Collective Construction by Termite-Inspired Robots

Collective Construction by Termite-Inspired Robots

A dissertation presented
by

Kirstin Hagelskjaer Petersen

to

The School of Engineering and Applied Sciences

in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of

Computer Science

Harvard University
Cambridge, Massachusetts

September 2014

Dissertation advisor:
Professor Radhika Nagpal

Author:
Kirstin Hagelskjaer Petersen

Collective Construction by Termite-Inspired Robots

Abstract

Construction usually involves careful preplanning and direct human operation of tools and material. Bringing automation to construction has the potential to improve its speed and efficiency, and to enable building in settings where it is difficult or dangerous for humans to work, e.g., in extraterrestrial environments or disaster areas. Nature provides us with impressive examples of animal construction: in particular, many species of termites build complex mounds several orders of magnitude larger than themselves. Inspired by termites and their building activities, our goal is to develop systems in which large numbers of robots collectively construct human-scale structures autonomously.

In this thesis I present TERMES, a system comprised of (1) A high-level control algorithm for decentralized construction of 3D user-specified structures using stigmergy, exploiting implicit rather than explicit communication; and (2) A complete physical implementation where three robots reliably assemble such structures using only local sensing, limited locomotion, and simple control, exploiting embodied rather than explicit intelligence. A major contribution of this work is the translation from abstract models to a real robotic system. I achieved this through careful co-design of algorithms and physical systems and of robots and building material, allowing passive mechanical features to minimize control complexity. To attain reliable performance without relying on costly high-precision sensors and actuators, I put an emphasis on error-tolerant control, making robots able to autonomously detect and recover from small errors. This work advances the aim of engineering collectives of robots that achieve human-specified goals, using biologically-inspired principles for robustness and scalability.

While our work is inspired by models of termite construction from the 1970s and 1980s, much is still unknown about how individual termites coordinate and respond to different environmental factors. To address this issue I developed methods and tools to enable high-resolution quantitative data collection on the behavior of individual termites engaged in collective construction in confined experimental arenas. This work advances our ability to study the termites which will hopefully lead to new insights on the design of robust autonomous systems for collective construction.

Table of Contents

Acknowledgements

First and foremost let me thank my friend and adviser, Radhika Nagpal. She not only trusted me to a position at Harvard, but also eventually convinced me to pursue grad school, and continued to mentor and support my work throughout.

I owe great thanks to Justin Werfel for his work on the algorithmic side of the TERMES project. We had many work-related arguments and did not always agree on how to do things, but in retrospect that was exactly what was needed to successfully implement a system that spanned so many different fields of research. Thanks to everyone who helped me develop my technical skills and for all of the ideas we pitched to each other during my years at Harvard; special thanks to Christian Ahler, Mirko Bordignon, Stan Cotreau, Kevin Galloway, Jim MacArthur, Nils Napp, Mike Rubenstein, James Weaver, and past and present members of the Self-Organizing Systems Research Lab. Nils, I hope to read much more about your tack on bio-inspired construction robots very soon.

Thanks to my architecture friends and colleagues for sharing their insights and providing the big picture for my research. And thanks to all of my friends and fellow researchers on the termite side of the project; Paul Bardunias, John Hallam, Lisa Margonelli, Erik Schluntz, and Scott Turner. Without your help and advice, crossing into the field of biology would have been terrifying!

I really appreciate all of my committee members, Dario Floreano, Radhika Nagpal, Margo Seltzer, Conor Walsh, and Rob Wood, taking their time to evaluate my work and giving great feedback. Thanks to Joanne M. Bourgeois, without whom I would have been buried in forms and paper work. And a special thanks to the Wyss Institute for Biologically Inspired Engineering for the tremendous support and nurturing environment they provided for my research.

Finally, a huge thanks to all my of my family and friends who have helped keep me moderately sane throughout my studies in Boston. I hope for many more great times with all of you.

Chapter 1. Introduction

The construction industry currently represents 13% of the world's GDP, and its share is expected to rise to 15% by 2020 [1]. Yet construction methods continue to revolve around careful preplanning and direct human operation of tools and material. The job is typically slow, dirty, and dull; people tend to get bored and make mistakes, and despite improvements over the last decades it remains a high risk industry. US construction workers report fatal injury rates nearly three times that of the average employee and are responsible for up to 19% of all work-related illnesses and injuries every year [2-3]. The industry suffers from a diminishingly skilled and aging labor force; it is difficult to attract youths to a sector which offers dangerous, typically short-term, jobs [4-6]. Introducing robots to the construction sector may not only replace humans in such hazardous environments, but also improve cost and efficiency, and enable specialized construction in settings that have traditionally been considered too impractical for humans to work in, such as extraterrestrial terrain or areas with extreme climate.

With the long term goal of bringing robots to real world construction sites, researchers are exploring a wide set of tools, from robot-aided assembly [7] to completely autonomous robots [8-11] and algorithms for both single- and multi-agent systems [12-15]. I am interested in the latter challenge: how to design collectives of construction robots, each much simpler and smaller than the structure they are building. With the field in its infancy we must first address the challenge of how to coordinate construction by large collectives in a scalable manner, and ensure that the means we develop to do so will work robustly in both theory and practice. I take inspiration from the incredible examples of collective construction found in the mound-building termites of Africa, Australia, and Asia. My dissertation involves foremost the design of TERMES, a system of autonomous construction robots assembling three-dimensional user-specified structures larger

than themselves in a controlled laboratory setting, and secondarily, research tools and initial hypotheses regarding the real termite construction process with the goal of inspiring future robotic designs.

1.1 Bio-Inspiration

"The first detailed account of termites was given to the Royal Society of London in 1781 by Henry Smeathman who had returned from a voyage to Guinea. It was said that his paper was received with some scepticism, which is scarcely surprising, for he described small insects that could build towers standing well above the height of a man." [16]

Nature provides us with many examples of how teams might cooperate on construction tasks, like hornero couples and families of beavers, but no scales are as impressive as those of the social insects where hundreds, thousands, and even millions of tiny individuals can produce functional global structures through their collective efforts (Figure 1.1.a.a-f).

The African genus of Macrotermes builds some of the largest mounds (up to 5m tall) relative to the size of the cm-scale individuals. The process has no central point of coordination and every termite has a limited range of sensing. Instead they rely on *stigmergy* [17] to coordinate through their shared environment; the modifications of the soil by one termite guide subsequent actions of other termites. For example, if one termite deposits soil to initiate a pillar, another loaded termite coming across this deposition may react by adding more material, creating a positive feedback mechanism until the pillar is completed. Typically when the pillar reaches a certain size, more pillars are initiated at a body-lengths distance, and eventually these are crowned by an arched ceiling that forms the beginning of a gallery. Despite their simple methods, termite construction brings even the most skilled masons to shame.

Termites normally reside close to their royal chambers and fungus gardens in their underground nest. The nest climate remains remarkably consistent throughout the seasons; indeed some researchers believe that the mound structure is mostly a by-product of the removal of wet soil during the rainy season to limit internal humidity. What is particularly striking about these natural

systems is their ability to cope with failure and adapt to unstructured environments. A fist-sized hole in a healthy mound will be detected in minutes and plugged within a couple of hours, and it is not uncommon to see mounds completely encompassing living trees.

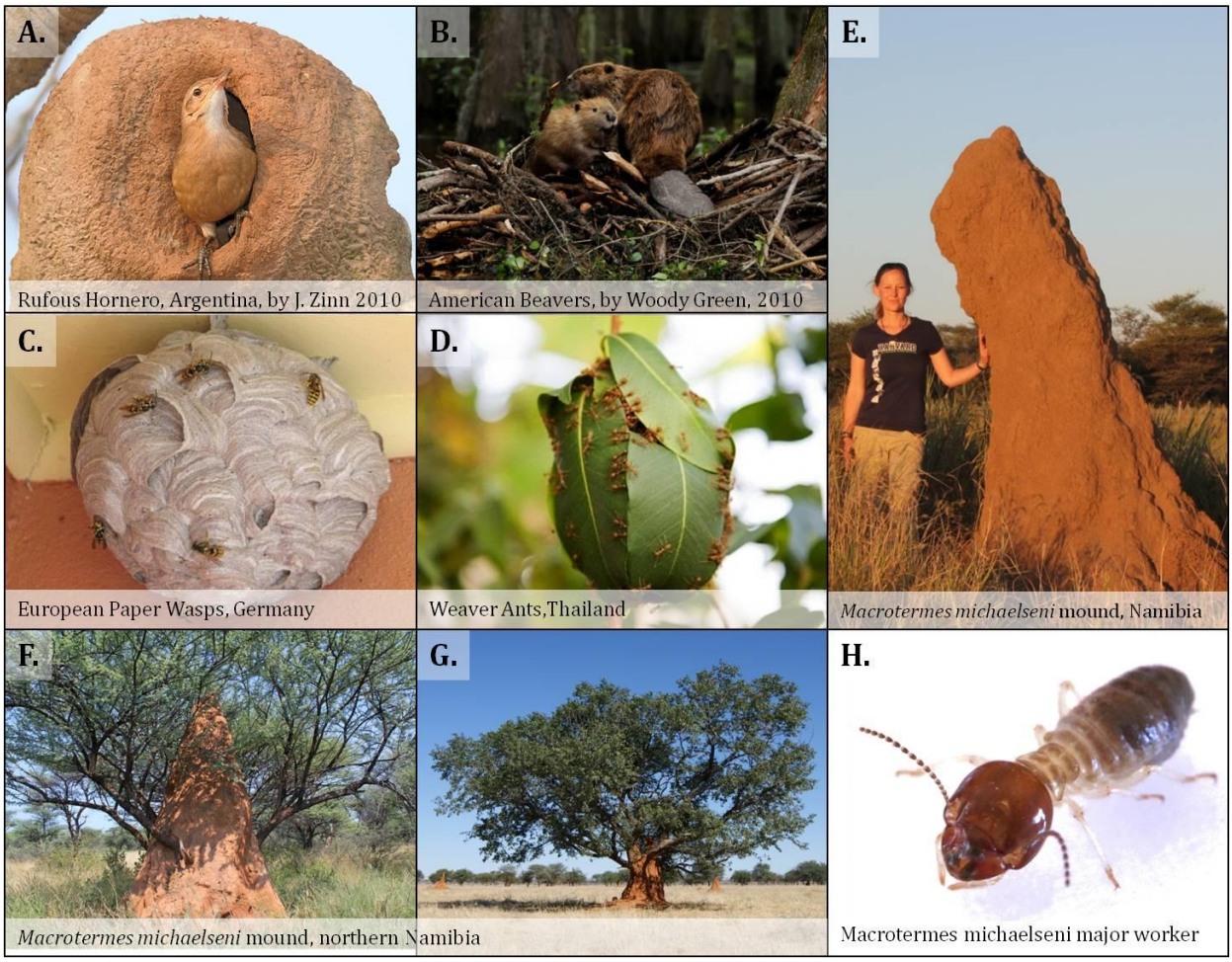


Figure 1.1.a. A: One of a pair of hornero birds with clay nest. B: North American beavers on their wooden nest; typically shared and maintained by 10-15 members of the family. C: European paper wasp nest with hundreds of individuals. D: Thai weaver ant colony with thousands of individuals. E-G: *Macrotermes michaelseni* mounds in Northern Namibia with millions of individuals. G: *M. michaelseni* major worker.

Though the skills of the termites are widely admired and provide a great source of inspiration for architects and engineers [18-20], very little is known about how the structure emerges. Termites are hard to examine, they prefer their underground dwellings and are very susceptible to disturbances. My collaborators are exploring the colony as a whole, the so-called super-organism [21], and others have suggested models based on pheromone templates and stigmergy [22].

The first half of my dissertation describes the design of a system of construction robots, which is inspired by termites and the way they perform decentralized construction with many simple, identical, climbing agents [23-25]. The second half focuses on new tools to study the decision making process of individual termites, whether all termites behave the same, and how, as a collective, they manage to produce functional global structures [26, 27].

1.2 Design Goals

Inspired by termites and trends in multi-agent robotics, several factors are key to the design of TERMES and in general any sustainable robotic system for autonomous construction.

Multi-agent, redundant systems with decentralized control. A collective of robots can offer many advantages over single robots [28]. They allow the option of an efficient parallel construction process, and without key-individuals the failure of one need not obstruct the entire system from succeeding. Traditional robots typically rely on centralized control, because it is logical and easy to design and allow efficient routing of all parts of the system, however, social insects are evidence that the opposite, decentralized control, allows much higher degrees of parallelism and error tolerance.

Restricted communication. Communication is often a bottleneck for systems with many agents, either because of a large number of messages leading to congestion and message loss, or more pressingly, because communicating across a large physical construction area crowded with building material may be impossible without the use of multi-hop communication which is hard to integrate reliably. To avoid this problem, we focus on collectives of robots that are limited to broadcasts, local knowledge exchange, or no direct communication at all. Instead we exploit stigmergy, utilizing the shared environment to pass information, thereby coordinating construction.

Local sensing and limited memory. Even traditional global sensors like GPS lack the accuracy needed for detailed construction and might fail when robots build structures that eventually shield themselves from communication range; instead sensors should be embedded on the robots. As we expect the structures to be much larger than the individual robots, it is infeasible for them to sense the entire structure at once; hence they should rely on local sensing, much like the blind worker caste of the termites does [29]. Furthermore, because many agents will be working on the same

structure, the information picked up by a single robot working on the structure is likely to become out of date as it leaves to find new material. Where limited memory might increase efficiency, e.g. by remembering what parts of the structure have progressed further than others, long term memory is not necessarily useful. Accordingly, the TERMES robots are restricted to local on-board sensing and exploit only limited memory to localize with respect to a seed brick.

Co-design, embodied intelligence, and simple mechanics. Co-designing abstract algorithmic agents and real physical robots ensures that conceptual ideas are indeed feasible. Co-designing all pieces in the physical system allows a highly optimized design, which in turn enables simpler and more robust robots to perform complicated behaviors by exploiting embodied intelligence [30]. The challenge of construction offers a unique opportunity to develop the world in which the robots operate, i.e. the structure and material. This can help limit the complexity of mechanics and control, decreasing the amount of sensory stimuli needed for the robots to make decisions, much like termites do when they create their own living quarters. Likewise, the TERMES algorithm through iterative design adheres to the restrictions of a physically implementable system, and passive mechanical features in the bricks simplify navigation-related sensing and control in the robots.

Error tolerance. Creating a large structure will require a large number of behaviors to be carried out flawlessly, however designing a robot that never experiences errors is virtually impossible. Instead, the focus must be on making every behavior error tolerant. A robot may fail at any one task (like climbing from one piece of material to the next), but should be able to detect such small scale errors and correct them before proceeding. Larger scale errors that can impede the behavior of future robots must be dealt with on an algorithmic level and may require different types of robots to intervene. The goal is not to make an error free system, but one that can tolerate errors that are bound to happen**.** The termite building process is messy and often experiences set-backs, yet the mound continues to grow through continuous reiteration of local construction.

Several factors trade off efficiency for robustness; centralized versus decentralized coordination, fast versus careful control with error detection, and more capable versus simpler hardware. Here I present a solution at one end of the scale; with no central coordination, no inter-agent communication, and identical robots with limited mechanical abilities, localized sensing, limited memory, and slow, but error tolerant control. Although the current system is based on homogeneous robots, future systems may benefit from the use of heterogeneity, where different robots are better suited for different tasks, much like different experts install bricks and insulation in human construction. The robots are intended for complete autonomous operation, however, future systems might very well benefit from a human monitor to correct rare, but fatal errors.

1.3 Contributions

The first half of this thesis presents TERMES, the algorithmic framework and the robotic implementation of a system for collective construction of three dimensional user-specified structures composed of square bricks (Figure 1.3.a). This was a joint project with Dr. Justin Werfel, a research scientist at the Wyss Institute, and my advisor Professor Radhika Nagpal. We designed the high-level algorithm and robotic test bed together, based in part on work I did for my master's degree at the University of Southern Denmark [31]. Within the TERMES project, the algorithmic framework was mainly developed by Werfel. I focused on the design and construction of a robotic system that fully implemented the algorithmic framework.

The TERMES project makes the following major contributions:

- A high-level control algorithm for decentralized construction of 3D user-specified structures using stigmergy, exploiting implicit, rather than explicit communication.
- A complete physical implementation where robots reliably assemble structures in 3D using only local sensing and simple control, exploiting embodied, rather than explicit intelligence.

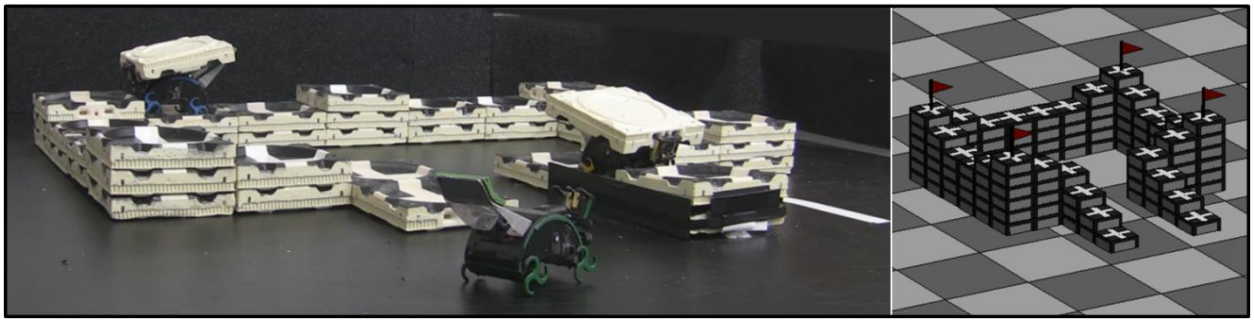


Figure 1.3.a. Collective construction of a castle-shaped structure, left: with real robots, right: in simulation.

Translation between abstract agents and real robots in the field of collective construction is rarely accomplished; we achieved this through a co-design of robots and algorithm; implementing hardware constraints in the abstract algorithm to ensure realizable physical robots. The physical implementation demonstrates a high degree of reliability, achieved by co-design of robots and bricks, as well as a strong focus on making robots able to detect and correct errors autonomously. Specifically, the implementation contributes the following:

- A wheg-based design for *locomotion* with simple control over level ground and up narrow structures of dedicated bricks guided by passive mechanical features.
- Reliable *navigation* relative to a structure of dedicated bricks based on simple sensory feedback on-board a robot.
- A single-actuator *manipulator* with passive mechanical features for a stable grasp and release of a dedicated handle in the bricks.
- A modular *software* architecture that fully implements the abstract algorithm on real robots, and allows for easy substitution of routines with changing hardware designs.
- A simple robot *algorithm* to construct a large class of structures, using the described physical system. (Joint work with Dr. Justin Werfel and Prof. Radhika Nagpal)

Future work will focus on improving reliability to enable construction of large structures with large numbers of agents, expand the class of structures admissible as well as examine non-deterministic outcomes. Creating a system where many robots cooperate to reliably build large scale structures without human interference over a long sequence of steps is a major challenge that we have yet to fully address.

The second half of this dissertation contributes exploratory methods and tools to enable studies of coordination of construction in termites. The main contributions are:

- Software to semi-automatically perform visual tracking of position and orientation in recordings of groups of unmarked termites in 2D arenas. (Joint work with Harvard undergrad Erik Schluntz).
- Software to semi-automatically label several behaviors related to construction in recordings of groups of unmarked termites in 2D arenas.

Representative results are shown in Figure 1.3.b. Using these tools I have found preliminary evidence for several hypotheses to serve as the basis for future, more rigorous experiments. I have investigated an arrestant property of fresh soil depositions in *Macrotermes michaelseni* and *M. natalensis* which may have been confounded with cement-pheromone in previous studies (joint work with Dr. Paul Bardunias at SUNY College of Environmental Sciences), differences in behavior between individuals of the same species, and differences between the two species. Furthermore, I have explored other methods and tools, including a 3D scanner to record detailed soil movement in experimental arenas ex-situ, and observation chambers to study the mound repair process in-situ.

Figure 1.3.b. Output from tools to study termites confined to 2D arenas, including semi-automated tracking, semi-automated behavior labeling, and recording of construction progress in 3D.

1.4 Organization of Dissertation

In Chapter 2 I review related work including technology for automated construction already employed in industry, research projects concerned with automated collective- and single-robot construction, as well as frameworks that attempt to directly mimic construction in social insects. Chapter 3 explains the algorithmic framework of TERMES and the biological corollaries in mound-building termites. It provides details of the algorithms and convergence proofs, as well as the class of admissible structures and how it performs compared to a centralized controller in different scenarios. Finally, the chapter concludes by suggesting future work and describing a few exploratory extensions to the framework. Chapter 4 describes the physical implementation of TERMES which is one of my major contributions to the project. The chapter provides a detailed review of what key factors led to successful implementation, and guidelines and pointers for future researchers in the field who may wish to co-opt this strategy. It describes mechano-sensory solutions to each of the three main challenges; achieving robust locomotion, navigation, and manipulation, and summarizes design of electronics and embedded software. It further describes the process by which new bricks and robots were fabricated; easy fabrication was key to the iterative cycle through which the hardware components were optimized. Finally, the chapter reports on the performance of the system as well as failure modes, future work, and some exploratory extensions to the current hardware. As mentioned, future systems for construction will be inspired in detail by how collectives of termites manage to coordinate construction of intricate pillars and roofs without centralized control. Chapter 5 gives an introduction to the Macrotermes termites, explains methods and tools developed to study them including visual trackers, behavior labelers and 3D scanners, and, finally, reports on initial discoveries made through these tools when exploring the effect of freshly deposited soil on termites. Chapter 6 concludes and suggests future directions.

With this work we have completed a cycle in the iteration between understanding the termites and co-opting the strengths of their system into robotic construction crews (Figure 1.4.a). Many such cycles are needed to comprehend all the features needed to construct large scale human-targeted structures autonomously.

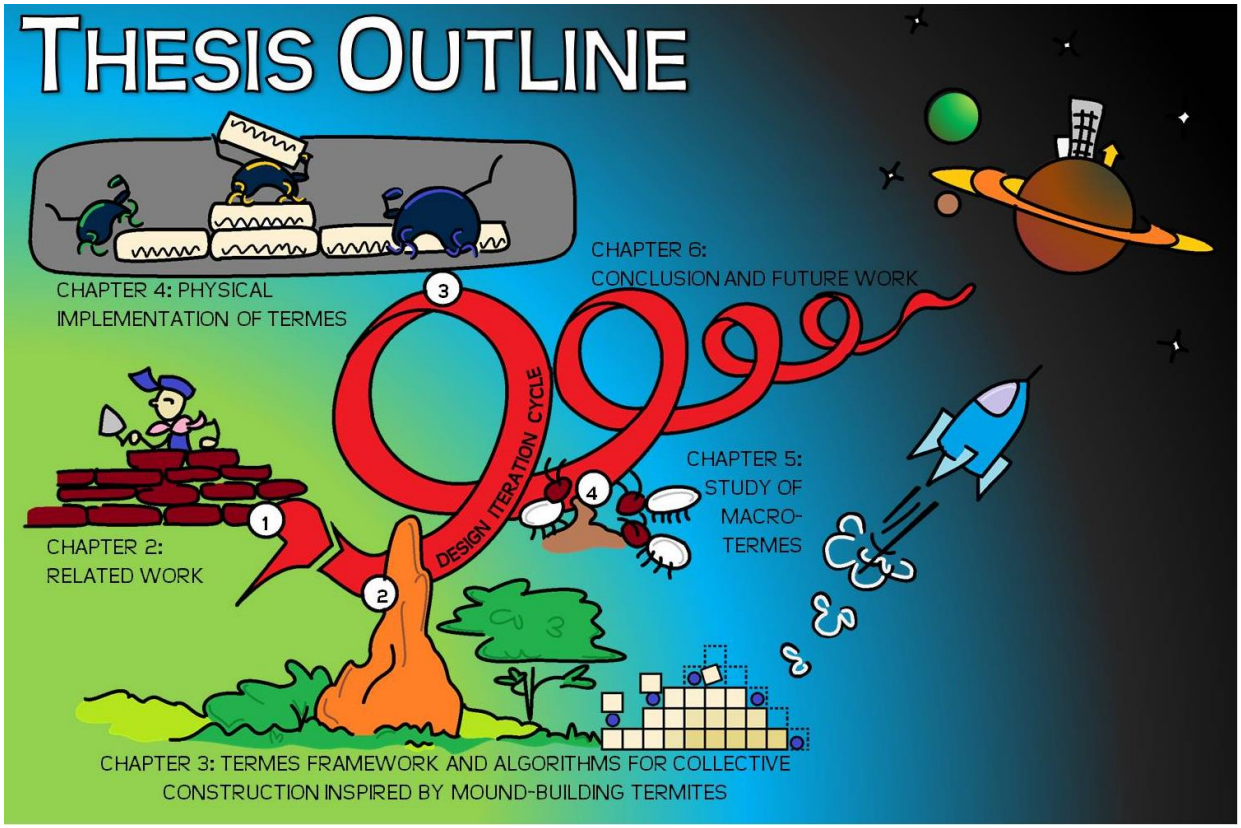


Figure 1.4.a. Illustration describing the thesis outline and iterative cycle between inspiration from termite studies and system implementation.

Chapter 2. Related Work

This chapter presents a broad overview of the field of automated construction, with special focus on multi-agent systems. There are many ways in which automation can improve construction; it can remove humans from dangerous conditions, improve efficiency, and reduce reliance on a diminishingly skilled workforce. Nonetheless, the only robots currently accepted by industry follow traditional construction methods. Moving from the current state of the field to full-scale automation is a huge challenge which requires a complete revolution in industrial methods, tools, and materials. New systems must prove their versatility and reliability, and possibly, to gain goodwill, first target construction of structures that are currently impossible with traditional methods; including specialized structures or construction in complicated settings.

The following sections review today's industrial pioneers, alternate methods for construction, collective construction, and research directly inspired by social insects. Research related to specific challenges of this dissertation, such as climbing, manipulation, Macrotermes, etc, can be found in their corresponding chapters.

2.1 Industrial Pioneers

Despite several robotic systems employed in the construction industry, automation has yet to be properly accepted and incorporated into standard usage, mostly due to high start-up costs. The biggest success stories rely on a combination of robot and human involvement. Broad Sustainable Building [32] use automatically fabricated modules to be assembled manually on-site. They have completed a 30-story high-rise in 15 days and have plans to erect a 220 floor tower in just 90 days. The Obayashi Corporation exploits both pre-fabrication and automated assembly, using a "Super Construction Factory" [33] to assemble steel members of a floor semi-automatically. When one floor is done, the entire factory hall is jacked up through an internal climbing system to commence work on the next floor. Their techniques have reduced manual labor up to 60% [34]. On a smaller scale, a recent robotic system SAM, for Semi-Automated Masonry, has been developed to build walls with brick and mortar [35]. The device operates on a horizontal track and requires a human operator to correct brick alignment and remove excess mortar. Other specialized robots have been used to reduce the cost on tile inspections, spraying of concrete, surface finishing, curtain wall installations, reinforcement, and welding [7] (Figure 2.1.a). These methods work well for direct incorporation into the building industry today, but still revolve around humans as the main work force. The following sections describe systems which seek to automate the entire construction process, to work in scenarios where direct human involvement may be less practical.

Figure 2.1.a. Sample of robot aided construction tasks in industry today.

2.2 Stationary Assembly

One approach to automated construction is to have stationary devices assemble structures. Khoshevis at University of Southern California argues for the use of contour crafting in construction. Contour crafting [8] and similar approaches [6, 36] are based on computer controlled layered addition techniques equivalent to large-scale rapid prototyping; e.g. concrete is poured in thin layers one at a time until the full 3D structure is finished. Its advantages include efficiency, construction with multiple materials at once, and the possibility of constructing specifically optimized building elements. An example of the latter could be to alter the geometry of air pockets in walls to achieve optimal insulation properties depending on where in the structure they are located. The major disadvantage is that for every use, one must erect a gantry device larger than the intended structure. Furthermore, the system suffers from traditional shortcomings like unstable properties of viscous material, requiring carefully supervised mixing of materials.

The Grasp Lab at University of Pennsylvania has proposed a robotic factory floor [37] with immobile agents assembling truss structures one layer at a time. When the first layer is finished the entire structure is raised and a new supporting layer is assembled underneath according to the desired final outcome. A simulated floor has been programmed with a sliding scale of central to decentralized control for efficient routing of material to the assembly site [38].

Common for both methods is that they utilize stationary devices the footprint of which must be larger than the final structure. The following section describes systems for collective construction which aims to utilize multiple mobile robots to expand structures far beyond the size of the initial footprint.

2.3 Collective Construction by Robots

Collective construction refers to multi-robot assembly of structures. A collective of robots offers the advantage of *parallelism*, having the structure progress on more than one front at once, *error tolerance*, the fault of one agent does not impede the progress of others, and the possibility of constructing on *large scales* relative to the size of individual agents. The systems discussed here mostly construct accurate user-specified structures, as are most common in human habitats.

The biggest 3D structures currently assembled by autonomous multi-robot systems were completed by teams of aerial robots [9, 39] with centralized control (Figure 2.3.a.d). Both systems exploit Vicon motion capture systems [40], providing detailed information about the position of all robots and material at all time. Aerial robots can build structures that are difficult for climbing robots like those of the TERMES system to maneuver on; however, they tend to have a poor payload-to-energy consumption factor and require complicated control and sensing to avoid collisions with each other and the structure they are creating. Ultimately, construction might be best achieved through the joint efforts of climbing and flying agents. Although the Vicon motion capture systems work well for positional feedback of many agents, it is unlikely that such high accuracy global sensors are available in real world construction scenarios, especially when the structures being built start shielding the robots from direct view. Centralized controllers offer the general advantage of efficient coordination of multiple agents, and the ability to guide robot movements on complicated structural elements, such as trusses [41, 42] (Figure 2.3.a.b), but suffer from a single point of failure and scalability issues as mentioned in section 1.2.

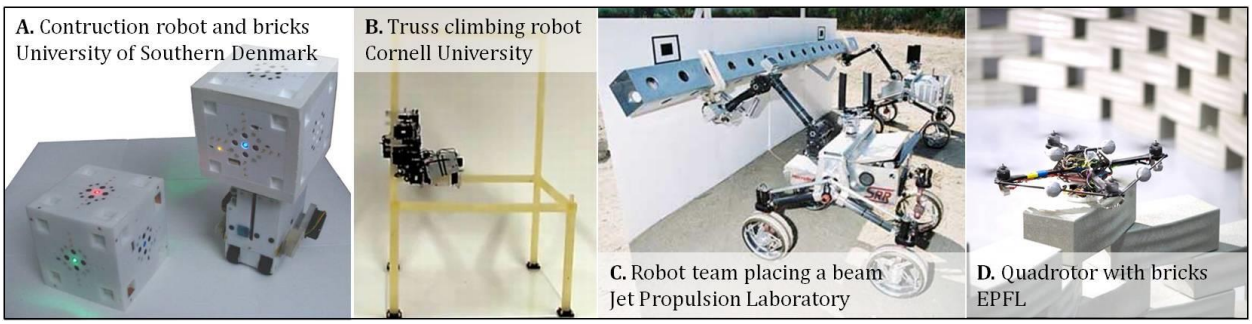


Figure 2.3.a. Four robotic prototypes designed for collective construction.

Other work focuses on decentralized coordination to optimize parallelism and minimize work imbalance by efficiently guiding robots and materials around on the structure [12, 13, 43, 44]. Many have proposed distributed control schemes for stochastic assembly of parts in 2D based purely on reactive agent behavior inspired by chemical reactions [14, 45-47]. Without a centralized controller, agents must coordinate by means of communication either directly or through the environment. More communication between agents typically improves efficiency with the trade-off of requiring more bandwidth as the number of agents scale. Systems will eventually reach a point where adding more agents are no longer beneficial. The TERMES system is at opposite end of this scale with no direct communication between at agents. Less communication requires more exploration to be done by individual robots, but allows for a more scalable system. Several projects examine the cost of communication [15, 48].

An alternative direction pursued is to transfer some of the intelligence and computation into the building material. Some systems, like TERMES, operate with completely passive material and leave all intelligence to the robots [10, 49, 50] (Figure 2.3.a-b); some add changeable id's to the building blocks for robots to alter to guide subsequent actions of future robots [51]; yet others divide up the intelligence and mobility between the building blocks and the robots, letting smart material guide dumb agents on the structure [31, 52, 53] (Figure 2.3.a-a). Finally, some researchers completely omit the design of builder robots, and envision all functionality directly in the

components of the structure [20], this includes the field of self-reconfigurable robots where robots can morph as needed to complete different tasks [15, 54]. If the final intent is to construct permanent structures, it is reasonable to aim for a system where the robots remain simple and robust, but are able to construct out of cheap passive material unlike the components of the self-reconfigurable robots.

All these systems remain research prototypes. None achieve the reliability and capability needed to build utilitarian permanent structures in the real world. Many of the algorithmic frameworks omit real world concerns, like robot movement constraints under gravity, or the difficulty involved in designing reliable robots able to maneuver, manipulate, and sense the structure they are working on. There is a large gap between abstract agents and real physical robots. The frameworks that have translated to physical multi-robot systems constructing in two- [45, 50, 51] or three- [9, 10, 31, 39, 53, 55] dimensions have shown limited success assembling at most a couple of pieces autonomously, or they rely on global high accuracy sensors that are hard to imagine in a real construction environment. The biggest issues with reliability appears to be with the mechanical interfaces between pieces of building material in the structure and between robots and material as they maneuver on top of it. The TERMES project [23-25] aims to improve reliability, by 1) co-designing the algorithmic framework and physical system to ensure that abstract agents are physically realizable, 2) co-designing robots and building material, allowing bricks and robots to be highly optimized to each other, so that complicated behavior can be achieved through simple hardware and control, and 3) incorporating error tolerance in every behavior.

The TERMES system is based on homogeneous fully autonomous robots; however, a future direction may be to have specialized robots monitoring and guiding others [56, 57] as well as the possibility for a human operator to take over control of a subset of agents temporarily to improve fault tolerance [58].

2.4 Collective Construction by Social Insects

Environmentally adaptive, and therefore error tolerant, construction directly inspired by social insects is a great incentive for automated construction in the real world. Entomologists and theoretical biologist have focused on how to built simple virtual agents to produce structures like those of wasps, ants, and termites. These structures share the property that the outcome is not necessarily fixed, but it adheres to some common rules, like comb size and pillar height, they are built by simple distributed agents able to sense only their local vicinity, and the agents use stigmergy to coordinate through their shared environment. Wasp-inspired algorithms are typically focused on a set of production rules where agents react to the local configuration of cells; termite-inspired algorithms are typically focused on agent reaction to local pheromone levels.

The model put forth by Karsai and Penzes in 1993 produce 2D comb structures, using stateless agents which randomly wanders the environment and can encounter a total of 10 different states all solvable with yes-no answers [59]. Advanced local patterns of construction, such as domes around queens and star-like chains, arise from the use of pheromone templates with termite-inspired agents [22, 60]. By further implementing diffusing properties of pheromone depositions, body size templates, as well as the effect of physical occupation of space due to construction site traffic, abstract termite and ant agents have built pillars, chambers, and isolated and intersecting tunnels in 3D [61-63]. Some work even suggests artificially evolving agent rule sets with a fitness function corresponding to the desired structure shape [64]. Recent work suggests the combination of wasp and termite inspired algorithms to improve performance beyond what they can achieve separately to build interwoven pillars and arches [65].

Roboticists have physically implemented several of these algorithms in limited laboratory settings. Parker (2003) designed robots that, like some species of ants, can clear out a circular nest

area using blind bulldozing techniques [66]. Others depend on templates and varying degrees of inter-robot communication and short term memory to build 2D linear structures [58, 67, 68].

Although interesting, the frameworks described here aim to estimate the global outcome of the collective, not necessarily model how individual termites actually coordinate construction. The secondary part of this thesis concerns the design of tools for direct observation of individual termites to enable studies of how global functional outcomes emerge from the local decisions of individual termites. These studies will hopefully inspire new frameworks for collective construction.

Chapter 3.
TERMES: Algorithmic Framework

We present a termite-inspired framework in which collectives of simple independent agents construct user-specified structures in 3D. The framework consists of two algorithms: an off-line compiler step and an agent rule set that combined allows agents to provably create a wide range of target structures exploiting local knowledge only. We have formalized the class of structures which can be built, and have further evaluated the efficiency of the TERMES system with decentralized control and completely independent agents compared to a centralized controller. We have also briefly explored framework extensions to encompass tasks in which the structure outcome is not pre-determined and how, with more capable agents, temporary staircases can be exploited to build otherwise unrealizable structures.

Section 3.1 motivates the ideas behind the TERMES framework as inspired by mound-building termites. Sections 3.2.1-2 describes the algorithms, section 3.2.3 gives an overview of the convergence proof, and section 3.2.4 describes the system and formalizes the class of admissible structures. Section 3.3 evaluates the efficiency of the TERMES system, and finally, section 3.4 describes possible framework extensions. A discussion of failure modes due to imperfect hardware when implemented in real life is reserved for Chapter 4.

The work in this chapter was published at Robotics: Science and Systems Conference (RSS 2011) [24] and at the Modular Robotics Workshop at the International Conference on Robots and Systems (IROS 2011) [25]; the final system was featured in Science magazine (2014) [23].

3.1 Approach

As discussed in Chapter 1 this work is inspired by mound-building termites which provide stunning examples of how to create functional structures with millions of individuals, each much smaller than the structure they are building. The TERMES project share several defining features with its natural counterpart including redundancy, scalability, minimalism, and error tolerance (Figure 3.1.a).

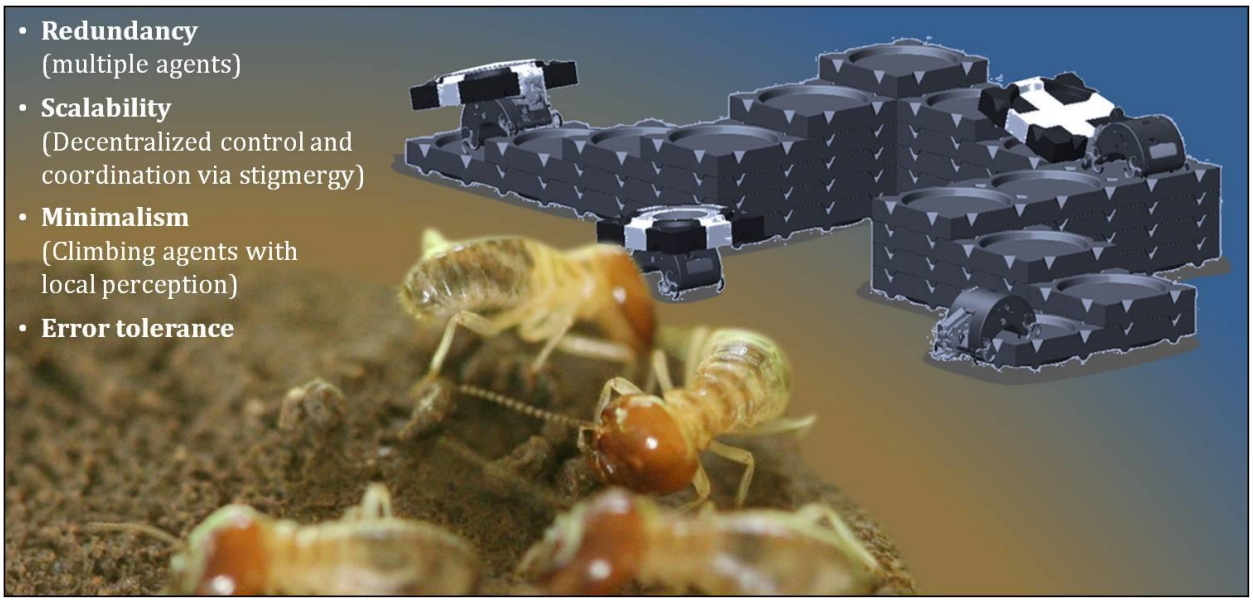


Figure 3.1.a. Inspiration from collective construction in termites.

Redundancy. Termite mounds are constructed by the combined efforts of millions of major and minor workers with no central point of coordination. Likewise, rather than one sophisticated robot, the TERMES system relies on many independent homogeneous robots with decentralized control to complete a structure. The key to success is redundancy; task completion is not tied to any specific individual, instead many agents work efficiently in parallel [69]. Centralized controllers with global knowledge of both structure and robots are able to solve problems in optimal time. However,

underlying communication and control schemes do not scale well; the controller is a possible single-point of failure that tends to be expensive and complicated to implement reliably, especially at a real-world construction site. Decentralized systems are forced to make decisions based on the local knowledge available to each robot and are generally less efficient. However, such systems naturally exploit parallelism [70, 71] and can be robust to the loss of single individuals.

Scalability. Efficiency is largely determined by the number of individuals that can work on the structure at once; in other words, the system must scale well. When many individuals restricted to local sensing work on the same structure, they need some way of coordinating their efforts. Using explicit communication between all agents is impractical in a large collective, especially in a setting crowded by solid building material. So far no research has indicated that termites use direct communication with each other to guide local building actions; instead they appear to use stigmergy to coordinate construction of mounds, tunnels, galleries and chambers. Although the TERMES system employs much fewer construction agents than a termite colony, coordination is still essential. The abilities of the robots are severely limited compared to termites; without coordination, material may be added in places that makes further construction infeasible or complicated by blocking the paths of future robots. Similar to termites, TERMES agents operate on the structure and react to the local configuration of material to determine if more should be added. With this approach, if an agent adds a brick to the structure, it will indirectly guide the actions of subsequent agents. By exploiting decentralized control and completely independent robots relying solely on indirect communication via stigmergy, computational complexity will remain constant as the number of agents increase.

Minimalism. The termites are far from simple, however, even the sophisticated physiology of a termite body is dwarfed by the ability with which millions reliably construct and maintain a mound so many orders of magnitude larger than themselves. The termites function only within the environment they shape for themselves, they are confined to climbing, they are limited to local

sensing, and seemingly not in need of communication to coordinate construction. Likewise, the TERMES system is designed with a minimalist approach; accurate navigation is performed only with respect to the structure on which the agents are working, they are confined to climb up limited inclines, they are limited to local sensing, and they require no direct communication. Simplicity helps, not only to make robust hardware realizable, but also to make it cheap and expendable so that large numbers of robots are actually feasible.

Error Tolerance. Finally, inspired by the termites, robot control was implemented with error tolerance in mind: Local construction by termites may appear messy, but through continuous collective effort they always manage to produce a functional high-level outcome. For details on the system implementation please refer to Chapter 4.

Rather than somewhat arbitrary soil mounds, the TERMES framework is concerned with construction of user-specified 3D structures. The current system is limited to construction with homogeneous square bricks by homogeneous robots; however, the algorithm may also extend to work with heterogeneous agents. In fact, future extensions of the system might benefit from a separate type of agents able to correct rare, but fatal errors of the builders.

3.2 Algorithms and Proofs

The TERMES system is able to construct a wide range of user-specified structures in 3D using a homogeneous set of square bricks and agents. The user provides a "blueprint" of the desired structure. This blueprint is processed off-line to produce a set of one-directional pathways over the structure, henceforth known as the *structpath*. The structpath is given to an arbitrary number of agents along with an internal rule set. Finally, collective construction can commence starting from a seed brick. This process is illustrated in Figure 3.2.a. The structpath changes with every new structure, however, the internal rule set of the agents is tied to their physical capabilities and remains the same. The structpath and agent rule set both help to direct the flow of agents over the structure as well as organize building activity so that the local information available to each agent is sufficient to build a provably correct structure.

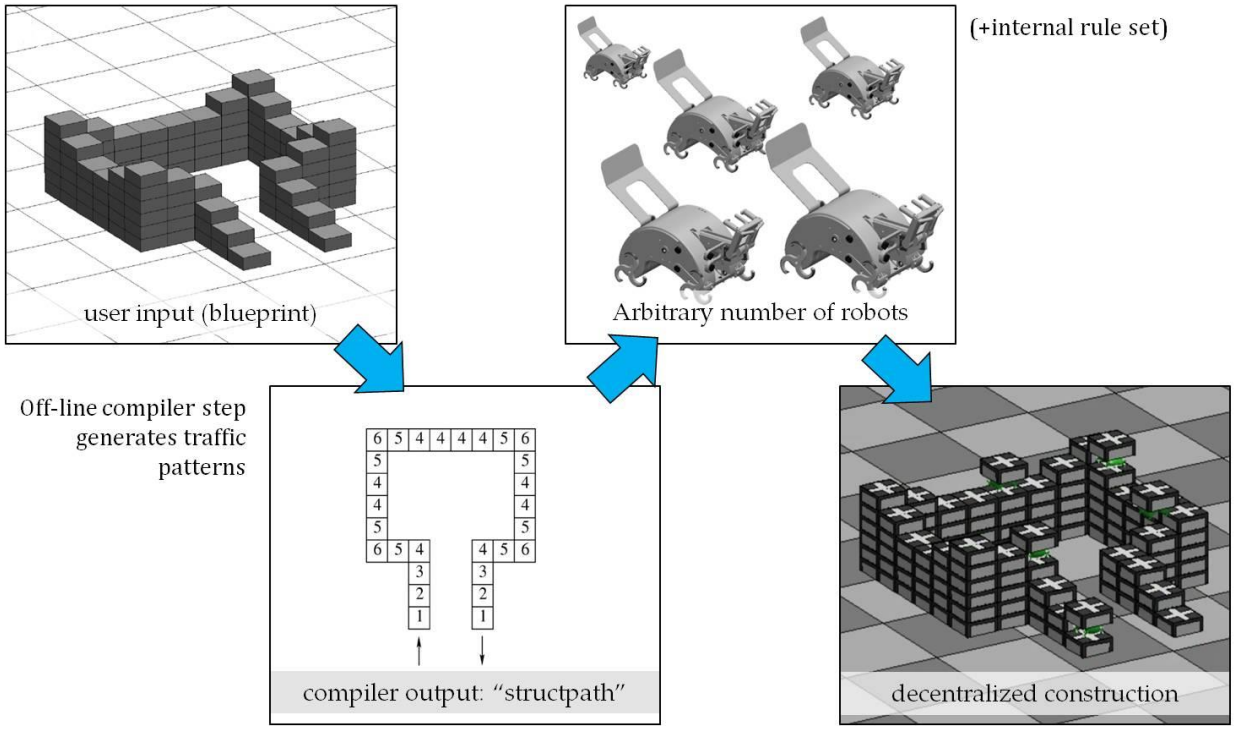


Figure 3.2.a. Overview of algorithmic framework.

Hardware constraints implemented at the algorithmic level:

- Agents are restricted to local perception to decide whether is it safe to attach bricks.
- Agents are only required to place bricks in the same horizontal plane on which they are standing and never directly in between two other bricks, which is physically difficult to accomplish.
- Agents are not required to remove bricks, i.e. all attachments are permanent.
- Agents never have to ascend or descend more than one brick-height at a time to get to other levels of construction.
- Agents do not need to communicate.

Figure 3.2.b. Hardware constraints, purposefully implemented at the algorithmic level.

An important aspect of this project is the strong tie between algorithmic development and hardware design. As mentioned in Chapter 2, many previous frameworks tend to underestimate the challenges associated with robust design of robots able to climb and manipulate material as well as navigate the structures they are modifying. Here, purposefully implemented hardware constraints at the algorithmic level ensure that translation to a physical system is feasible (Figure 3.2.b).

The effect of these constraints is discussed in more detail in section 3.2.1 and 3.2.2. The local rules of the algorithm are general and it is possible to generate a large class of structures with any robot design that complies with those constraints. The class of feasible structure is discussed in section 3.2.4.

3.2.1 Structpath Compiler

A system where agents freely add missing bricks will quickly produce deadlock, such as un-climbable cliffs or holes in the intermediate structure that are impossible to fill. The TERMES system exploits the structpath to help direct the flow of agents over the structure so that bricks are

added in an orderly fashion, and furthermore prevents situations where agents meet head on and have to determine who gets the right of way. A structure can have many viable structpaths; however, because all agents must agree on the same one, the structpath is generated in an off-line compiler step and given to each agent before construction commences. Examples of structpaths are shown in Figure 3.2.c, composed of a static 2D representation of the user-specified target structure, with a number on each site annotating the final stack height, and arrows indicating travel directions between bricks.

The structpath compiler is responsible for taking the user input and (if possible) generating a structpath; a viable assignment of one-directional paths through the structure which upholds the restrictions mentioned in Figure 3.2.b. Situations in which agents have to place bricks directly in between two other bricks are prevented by enforcing a *row rule*; if a brick is added in a horizontal layer the structure must grow from that point outwards (Figure 3.2.d). As illustrated in Figure 3.2.d.c loops in the structpath often lead to a breach in the row rule and are pruned from the search tree of the compiler.

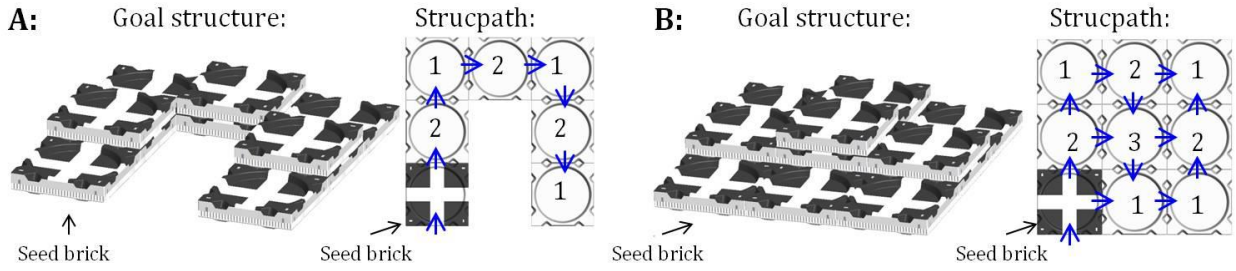


Figure 3.2.c. Examples of structpaths. A is a structure with a single path through it. In B agents can choose multiple paths through the structure starting from the seed brick. The number at each site specifies the desired final stack height.

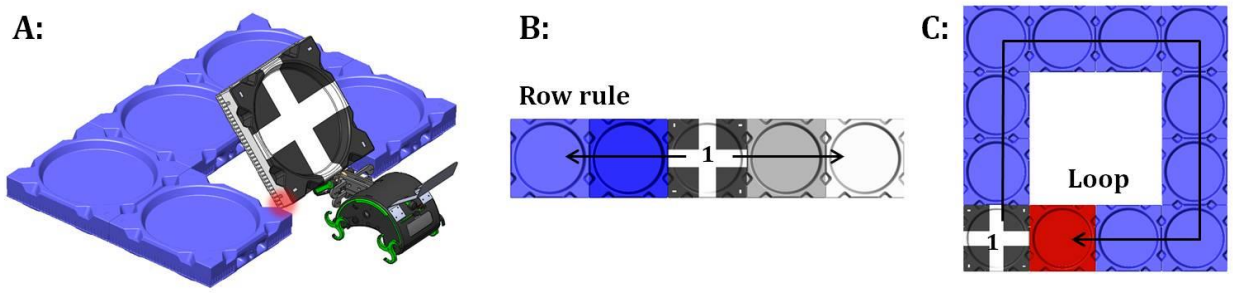


Figure 3.2.d. Illustrations showing the implication of agents not being able to place bricks directly in between two other bricks (A). The solution is the "row rule" (B), the structure must grow consecutively outwards from any new brick placed in a row. Loops in the structpath (C) often leads to conflicts in the row rule and are all pruned from the search tree of the structpath compiler.

Figure 3.2.e provides details and an example of how the structpath compiler works. Recursively, the compiler picks a site with at least one labeled and one unlabeled edge and assigns a direction from that point outwards to the end of the row. Solutions which contain loops or sites with only outgoing or incoming directions are pruned from the search tree. Depending on the structure there may be no solution at all or many possible solutions, however this type of depth-first search is guaranteed to be able to check all possible labels in finite time [72]. The compiler runs until all edges have a direction assigned and returns the first solution found, or terminates if no solution exist.

Structpath compiler. Given a target structure with a designated seed brick this pseudocode will return a valid structpath where all edges are labeled with travel directions (arrows), or identify that no solution exists.

1: **initialize** (seed site has an incoming arrow where agents enter the structure; all other edges are unlabeled)
2: **while** any sites have unlabeled internal edges **do**
3: choose a site with at least one labeled and at least one unlabeled edge
4: choose one unlabeled edge of that site
5: add arrows in a straight line from initial site to the end of contiguous row
6: **if** current graph has loops
or any site has all edges labeled without at least one being a traversable incoming arrow and (unless specified as a site from where agents can exit the structure) at least one being a traversable outgoing arrow then this labeling is untenable:
7: eliminate it from the search tree
8: **else**
9: go to line 2

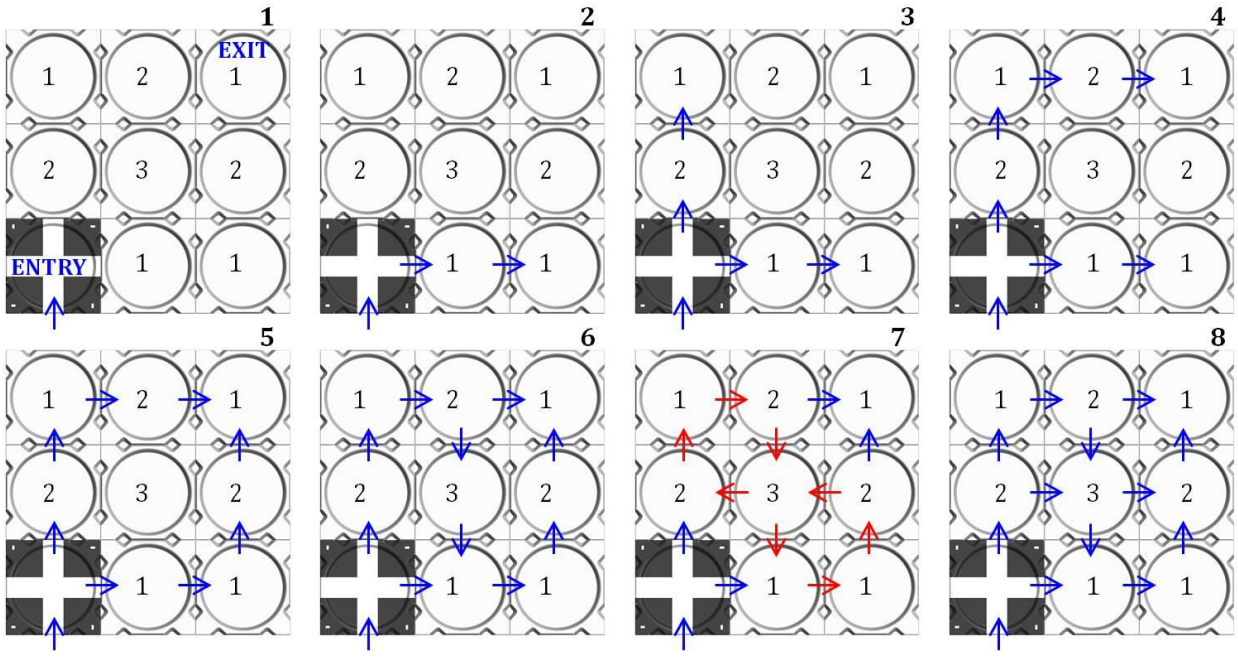


Figure 3.2.e. Top: Pseudo-code for the structpath compiler. For an arrow to be traversable the difference between the final heights of the two sites it connects must be no more than one. Bottom: Example of a structpath compilation based on the target structure shown in Figure 3.2.b.b. Step 7 produce loops in the structure, these are automatically pruned from the search tree, and travel directions are projected in the opposite direction instead.

Several heuristics and tools can be applied to improve the efficiency of the compiler. Rather than picking sites at random the compiler can be biased to pick unlabeled sites which are closer to the seed brick first. Structures which are impossible to create (discussed further in section 3.2.4) can be identified in a preprocessing step and rejected. Branches of the search tree which have proved infeasible, e.g. because of loops, can be memorized so that they are not compiled more than once. The complexity of the solution depends on nontrivial properties of the structure. Generally linear, single-path, structures are much easier to compile that multi-path structures. A brute-force approach can solve the problem in time exponential to the number of sites: each site has four edges which can be assigned in one of two directions. However, because there can be no loops and every site must have at least one outgoing and incoming edge large areas of the search tree can often be pruned and typical performance appears much faster than the worst-case scenario.

3.2.2 Agent Algorithm

The agent algorithm is independent of the goal structure: it is a purely additive algorithm which restricts the order of brick attachments to prevent situations that could hinder future progress, and it is compliant with agent capabilities, including local sensing, climbing at most one brick at a time, and only attaching bricks at the same level at which they are standing (Figure 3.2.f).

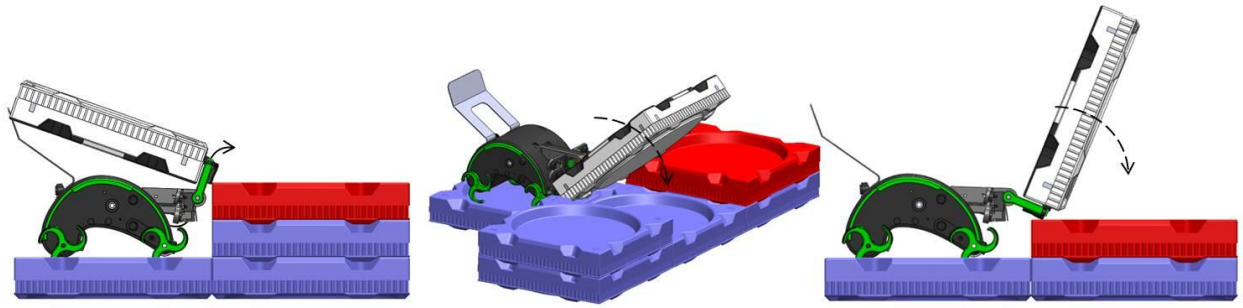


Figure 3.2.f. The agent algorithm prevents un-climbable cliffs, and the need to place a brick directly between two other bricks as well as at a different level from where the agent is standing.

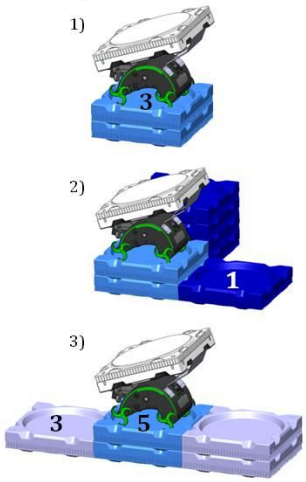
The agent algorithm is shown in Figure 3.2.g. It involves agents obtaining new material at a brick cache, finding the structure, entering though the seed brick and then traversing the structure according to the structpath. The seed brick provides a unique landmark that ensures spatially coordinated construction because all agents enter through it. Agents keep track of their position in the horizontal plane as they move over the structure grid, by turning ninety degrees left/right and moving straight between bricks, and store information about the height configuration of the last few bricks passed. If an agent encounters a site where a brick is desired, but missing, it will check whether it is okay to attach the brick based on its position in the structpath and the local configuration of bricks. If okay, the agent moves past the site turns around and places the brick. Because it is restricted to local sensing the agent must pass over the site of interest to detect if a brick is missing or not. If at any point the agents detect another agent nearby they hold their place until the other agent is no longer perceptible. For an agent to assess whether or not it is safe to attach a brick at a site in the structpath it must check the final desired stack height at that site and of sites leading to and from it (referred to as parent and child sites respectively).

Figure 3.2.h gives an example of how a structure will emerge, as a combined result of the agent algorithm and the structpath compiler, in growing levels of staircases.

Agent algorithm Agent control loop for predefined structures. H_i is the desired final height of the stack of bricks at site i, h_i the current height of that stack; the agent's current location is designated site 0. "Parent" and "child" sites refer to sites leading to and from the site respectively; "next" sites are children with $|h_i - h_0| \leq 1$ (i.e., the step to reach them is traversable). The exit off the structure is defined as a child site of the neighboring bricks.

```
1: loop
2:    get brick from brick cache
3:    go to structure
4:    follow perimeter counterclockwise until entry point
5:    climb onto structure
6:    while on structure do
7:       move to any "next" site
8:       update position
9:       if holding brick
         and h0 < H0 [1]
         and  for all parent sites i: (hi > h0 or hi = Hi) [2]
         and for all child sites i: (hi = h0 or |Hi - H0|>1) [3] then
10:         move to any "next" site
11:         update position
12:         attach brick at site just vacated
13: interrupt: if robot close in front then
14:    wait till gone
```

Satisfaction of line 9:



Violation of line 9:

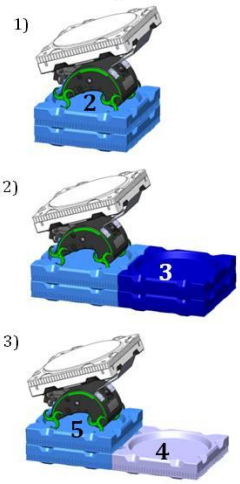


Figure 3.2.g. Pseudo-code for the agent algorithm. At any site, *i*, the agent addresses whether or not it is safe to attach a brick. Those safety checks are shown in line 9 and situations which satisfy and violate these conditions are illustrated to the right. In the illustration, only relevant fragments of the structure are shown. The number at each site specifies desired final height of that stack. Dark and bright colored bricks are parent and child sites of *i* respectively.

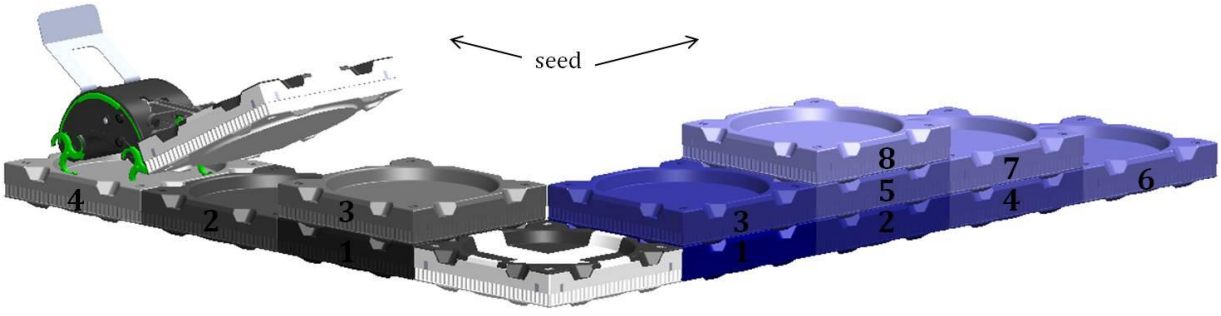


Figure 3.2.h. Illustration showing the order of construction, notice the growing levels of staircases as bricks are added in compliance with the structpath and the agent algorithm. Darker colors are placed before lighter ones, but the branches grow independent of each other.

3.2.3 Proof of Convergence

This section gives an overview of the convergence proof for the two algorithms (agent and structpath compiler). This proof was done by Justin Werfel, and is only outlined here for the sake of completion. For full details please refer to [23]. The proof is divided into four steps:

1. Agents never build configurations of bricks that prevent their physical progress along any part of the structpath.
2. Agents never build configurations in which they cannot physically attach bricks at any sites where bricks are desired.
3. Agents never build configurations in which they could physically attach additional desired bricks, but are prohibited everywhere from doing so purely by the logical (not physical) restrictions of Algorithm 2.
4. Different independent agents will not attach bricks at mutually conflicting sites.

Take a single-path linear structure as a simple example; it is known that all sites in the structure can be reached from the entry to the exit point, i.e. the final height of all sites differ by a maximum of one brick (Algorithm 1, Line 6). 1) The only thing that can impede physical progress along the structure is the creation of an un-climbable cliff. According to Algorithm 2 Line 9.2 an agent will only place a brick at site *i* if the parent site is already taller than the current height of *i* and it will not be able to place a brick unless the child site is at the same height as *i*, therefore a cliff will never be created. 2) The only thing that can prevent physical attachment of a brick at site *i*, is if an agent has to attach that brick from a different level from where it is standing; i.e. another brick was added prematurely and left a hole in the structure. This situation will not occur because the parent site to the wrongly attached brick will have been at a height equal itself, line 9.2 is violated. 3) Because

agents never create cliffs and because there is a traversable path through the structure, a site always exists where a brick can be added. 4) In single-path structures all bricks are added in a fixed order, therefore different agents cannot place bricks at conflicting sites.

The proof for multi-path structures is more complicated, but it follows the same guidelines, except now agents must check multiple parent and child sites, to enforce the row rule and to not create un-climbable cliffs, before adding a brick. Because the structpath guarantees that every site in the final structure has at least one traversable incoming and outgoing edge, bricks can be flanked by un-climbable cliffs from one side as long as they are traversable from the other sides. Furthermore, because of the row rule, agents might take paths through the structure along which bricks cannot be added because of logical constraints (parents and child sites have not yet reached the appropriate height), however, eventually as rows are extended construction at all sites will become possible. Although simultaneous addition of bricks is possible in a multi-path structure, all decisions on whether or not it is safe to add bricks are based on local configurations only, therefore multiple agents cannot place bricks at mutually conflicting sites. Figure 3.2.i gives an example of how construction progresses.

The interrupt routine of Algorithm 2 ensures that agents do not bump into each other. The one-directional paths on the structure limit the frequency with which agents encounter each other head on; when they do, a randomized time-out and the fact that the independent agents are not synchronized will prevent any deadlock issues.

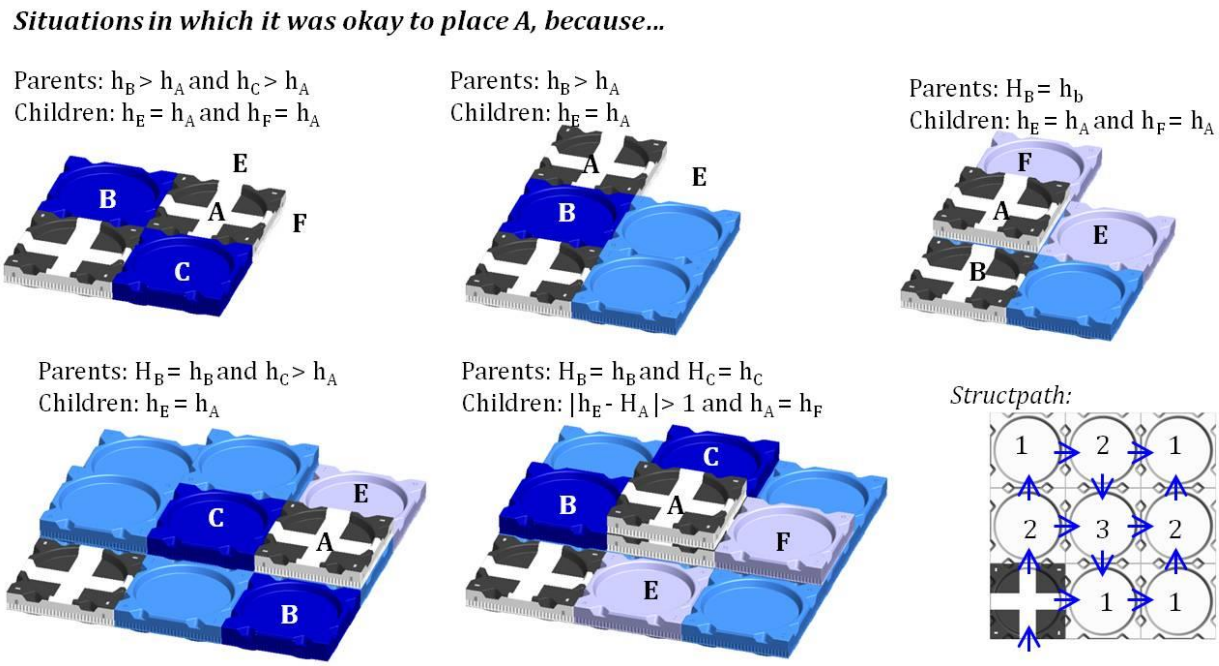


Figure 3.2.i. Examples of how line 9 in the agent algorithm (Figure 3.2.g) affects how the structure in the lower right corner is assembled. Bricks in dark colors annotated B-C are parents of A; bricks in bright colors annotated E-F are children of A.

3.2.4 Admissible Structures

Five constraints form the major boundaries of what can be built with the TERMES system:

- Square bricks can only form lattice-based structures.
- All bricks must be supported by a stack of bricks underneath, i.e. no overhangs.
- While on top of the structure agents cannot travel alongside (or turn) immediately next to a wall taller than the height at which the agent is holding the brick.
- There must be an entry and at least one exit point on the first level of bricks on the exterior boundary of the structure.

- Every site in the goal structure must be accessible by the agents, meaning that a path through the corresponding site must exist which starts and ends on the first level of bricks and changes by no more than one brick height between sites.

Figure 3.2.j illustrates some of the many types of structures which fit within these boundaries: linear, branching, and solid structures.

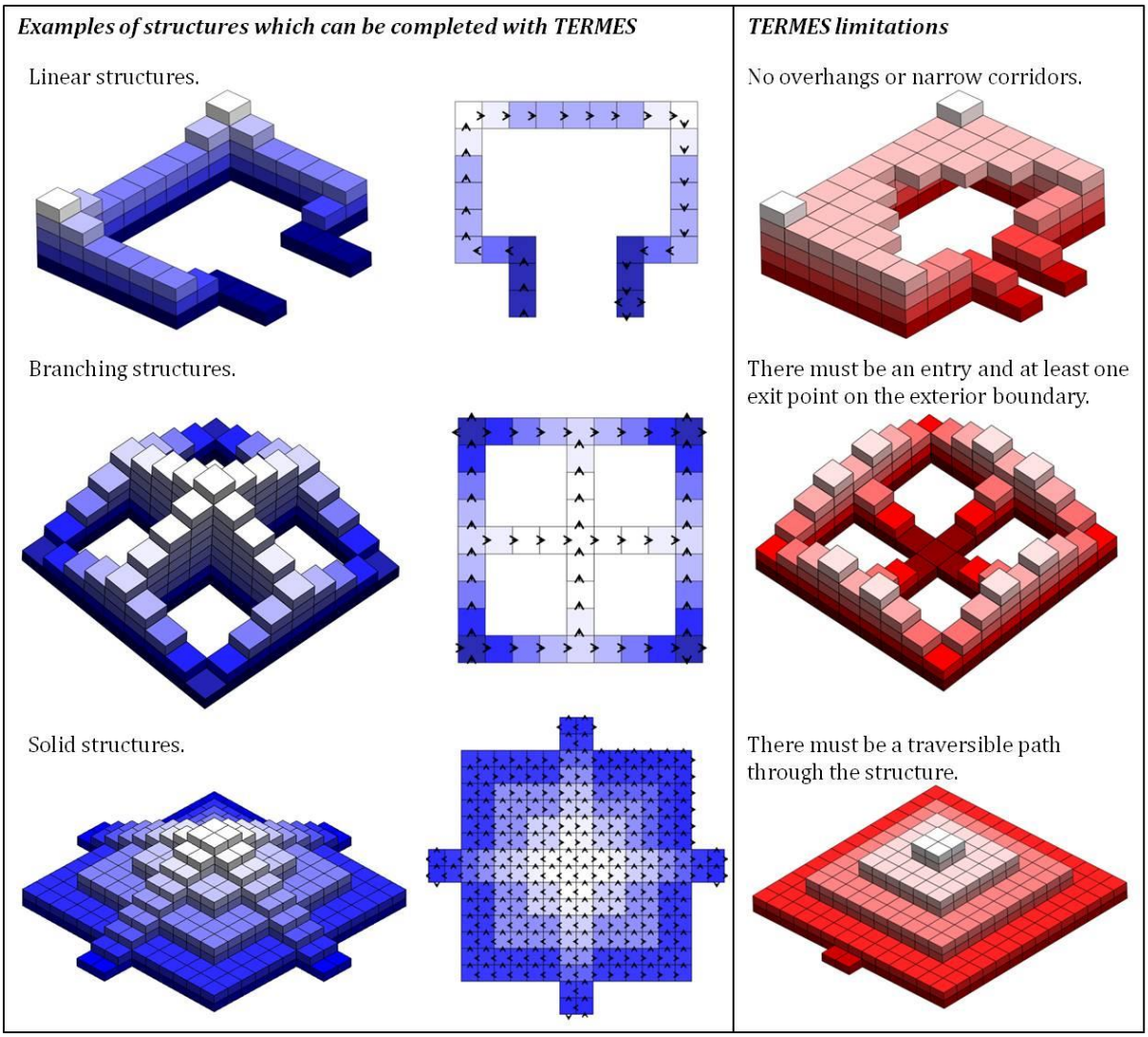


Figure 3.2.j. Illustration showing the class of structures which can be completed with TERMES, as well as some that cannot.

3.3 Performance

This section reviews strengths and weaknesses of a system like TERMES comprised of completely independent agents. It is important to remember that regardless of the type of coordination the efficiency of TERMES is first and foremost impacted by hardware constraints. Robots can only climb one brick at a time, and can only place bricks at the same level as the one they are standing on. The structure only has a single entry point and brick cache, causing a bottleneck for robots. With more capable hardware these constraints could be loosened to greatly extend the use of the system independent on the type of controller. However, as explained in Chapter 2, such advanced systems have yet to be robustly implemented in hardware. For the TERMES system, the hardware constraints presented here were all consciously chosen, not only to make the physical system realizable, but also to help limit the need for global knowledge. The following section reviews what a system like TERMES could gain from global knowledge, see Figure 3.3.a.

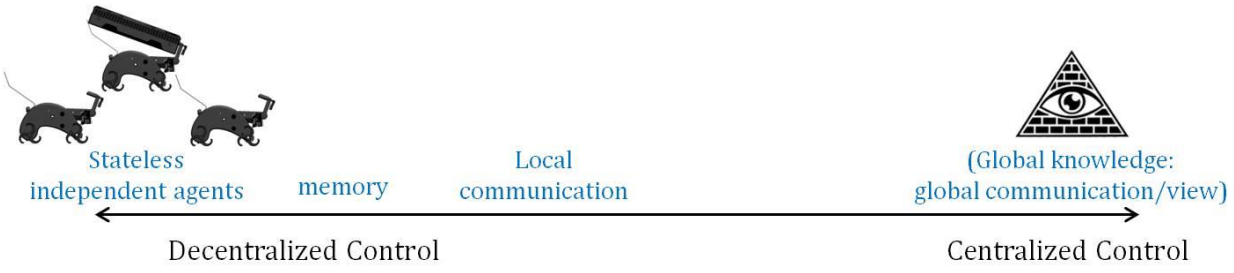


Figure 3.3.a. Sliding scale of knowledge, from an omnipresent centralized controller to a decentralized system with completely independent agents. Agent knowledge may be increased by exploiting memory and communication.

A centralized controller can optimally route agents through any structure based on global knowledge of structure progress and the position of all robots. The trade-off is added hardware complexity: the centralized controller, global feedback system, and global communication links to every robot must be very robust to outweigh the risk of adding a single point of failure to the system. In decentralized systems agent knowledge is the limiting factor for optimal construction of multi-path structures. The TERMES algorithm is designed for agents with limited memory completely independent of each other. The agents use memory only while on the structure to keep track of their position in the structpath and the height configuration of nearby bricks; i.e. agents do not remember the state of the structure they passed over; this is liable to become stale information as other agents modify it anyway. As agents can only perceive and remember the local configuration of the structure and are only able to communicate with nearby agents (Figure 3.2.b), there is no reason for them to share information. Later in this section I discuss how the completion of certain types of target structures might benefit from agent memory and local knowledge sharing.

Given the current hardware constraints, it is difficult to generalize if and how much a multi-agent system can gain from a centralized controller, over a system with completely independent agents. Here, I examine three classes of structures including single-path structures, branching-path structures and structures with paths that branch and join (Figure 3.3.b). For simplicity, each structure will only be one brick tall. Each scenario is compared based on the amount of energy it will take to build. In this abstract model, there will be an infinite number of robots and each robot starts on the seed and spends one energy unit per brick it passes over, the distance to get back to the seed brick is omitted. This is similar to the real system where energy consumption is dominated by the effect of robots having to travel over the structure to get to where bricks can be placed.

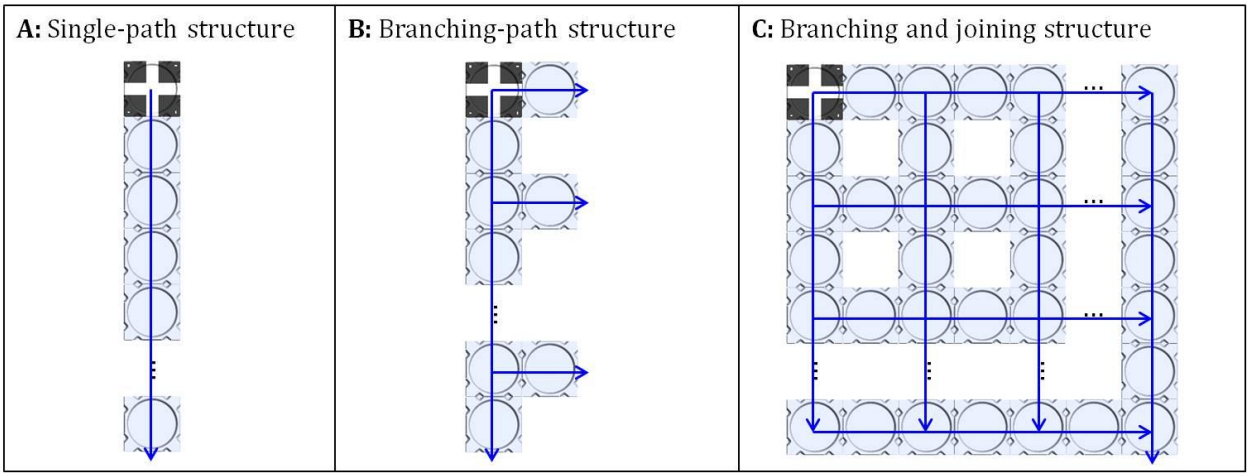


Figure 3.3.b. Illustrations showing the types of structures under evaluation.

The first structure (A) is a single-path structure with n bricks. Because there is only a single path through the structure all robots following the structpath will come across a point where a brick can be placed: A centralized controller offers no advantage on this class of structures.

The second type of structure (B) includes branches, in this case the number of branches scale linearly with the number of bricks. A centralized controller will know when branches of the structure have been completed and therefore know when to direct robots in other directions. The minimum energy required to build these structure is:

$$E_{MIN} = \sum_{x=1}^{n-n/3} x - 1 + \sum_{x=1}^{n-n/3} \begin{cases} if\ x\ odd & x \\ else & 0 \end{cases}$$

where n equals the number of bricks; the first term is the path length to complete the main structure; and the second term is the path length to complete each branch. With decentralized control each robot may chose a branch at random, in which case it will become harder and harder to complete the branches furthest away from the seed; the energy spent grows exponentially with the number of bricks/branches in the structure. A smarter system, still utilizing completely independent agents, might distribute the probability of entering every branch equally (Figure 3.3.c.a). Figure 3.3.c.b shows the performance of both systems for simulated trials with 100 replicas.

The amount of wasted energy of the smart system still increases (though much slower) with the size of the structure. If the number of agents is limited compared to the size of the structure (so that each agent will have to pass over the structure many times), a simple way to improve performance is to let the agents remember what finished branches they have already visited. Initially when all agents are clustered around a small structure and often come in contact, local information sharing could also be helpful.

The third class of structures includes paths which branch and join (Figure 3.3.b.c). A centralized controller will route agents through such structures in an optimal manner; The minimal energy spent, i.e. the sum of the shortest path to each brick, is simply the sum of the distance along the x and y axis to each brick minus the initial position of the agent on the structure:

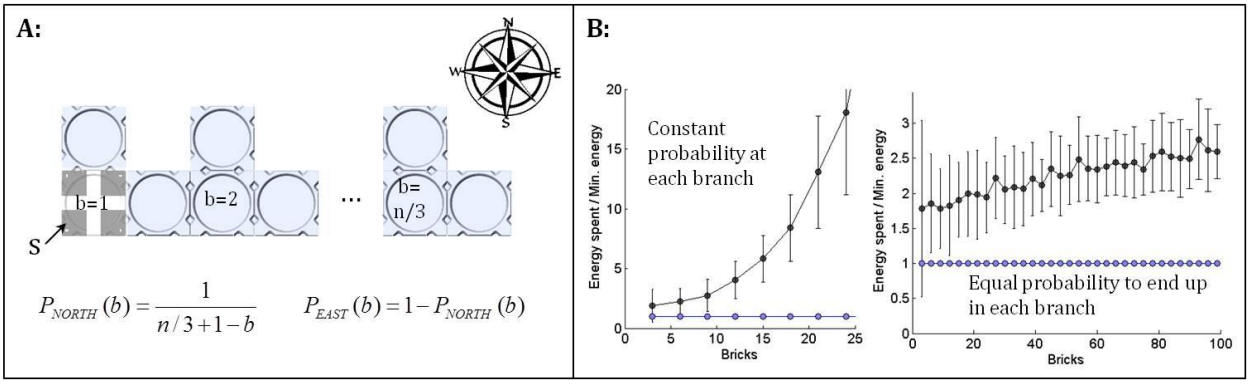


Figure 3.3.c. System performance on the type of branching structures shown in Figure 3.3.b.b. A: Shows how the probability between branches can be scaled so that agents have an equal chance of entering any branch. B: Shows how performance scales with the number of bricks when agents have either a fifty-fifty chance of choosing any branch or a probability scaled to the number of branches. The energy spent is normalized with respect to the performance of a centralized controller, i.e. the minimum energy needed to complete the structure (also shown in blue). The simulations were done with 100 replicas each.

$$E_{MIN} = \sum_{x=1}^{2 \cdot no.of\ squares+1} \sum_{y=1}^{2 \cdot no.of\ squares+1} \begin{cases} if\ brick\ at\ (x,y) & x+y-2 \\ else & 0 \end{cases}$$

The performance of the TERMES system, without memory or local knowledge sharing, is shown in Figure 3.3.d.b. Again, the system gets increasingly worse with the size of the structure. The strategy here was simple: at any branch choose between options with equal probability. The best decentralized agent strategy is not obvious for two reasons. First, in some cases joining paths will increase the chance to find a specific empty spot, see Figure 3.3.d.a. Second, because of the row rule, it might be advantageous to bias agents to build along the outside edges of the structure in the beginning, and then slowly migrate inwards. Such a feature might be implemented using memory of how many times an agent has traversed the structure, or by local memory sharing. With a large number of interwoven paths it is unclear how much the system would improve.

Implementing a robust centralized controller is a complicated and costly affair. For single path structures centralized controllers offer no advantage at all. For multi-path structures, decentralized controllers perform almost as well as centralized controllers in the beginning of the process (because initially finding a place to attach a brick is easy). Later in the process, centralized controllers do much better because they can easily find the path through the structure that leads to the remaining empty spots. This tendency is illustrated in Figure 3.3.d.c for a 341 brick (=100 square) structure. The decentralized system discussed previously completes 50% of the structure in only twice the optimal time. Beyond 60% payback starts diminishing. Notice the effect of the row rule marked with blue arrows: the sudden spike in progress each time a brick is placed in a branching point on the edge. As stated earlier, a strategy based on memory or construction time to let robots focus on the edge first and the center of the structure later may improve performance.

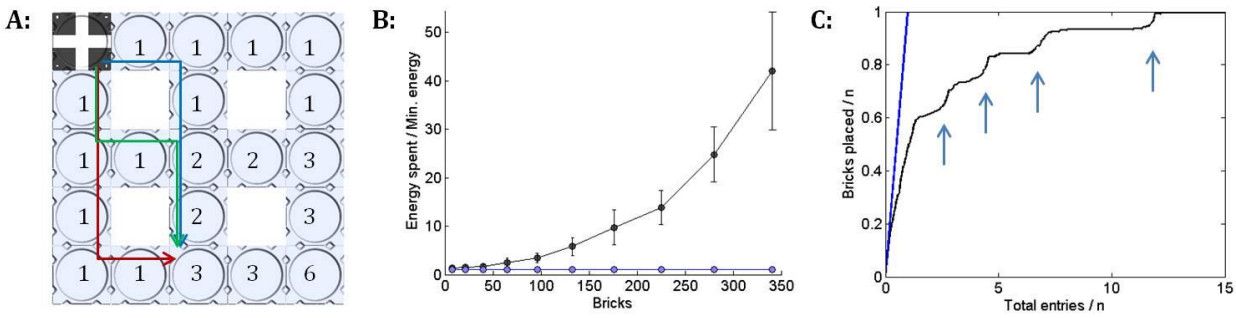


Figure 3.3.d. System performance for structures with paths that branch and join, see Figure 3.3.b.c. A: Shows how some places in the structure are more likely to be reached than others if the probability for choosing any branch is constant throughout the structure. The number in each square denotes the number of paths lead to it. B: Shows how performance scales with the number of bricks up to 341 (=100 squares). The energy spent is normalized with respect to the performance of a centralized controller, i.e. the minimum energy needed to complete the structure (also shown in blue). 100 replicas where done for each structure size. C: Example of structure progress over the number of agents which have entered the structure, normalized to the total number of bricks. The performance of the centralized system is shown in blue. The arrows indicate sudden bursts of activity in construction caused by depositions made to branching points on the edge of the structure; otherwise progress is slowed down because of the row rule.

The optimal approach depends on what parameters are of importance. If completion time is important, the goal is multi-path structures, and robot hardware is expensive, it may pay off to rely on a centralized controller. However, if robots are cheap and expendable and/or time is less important compared to cost and robustness, a decentralized system offers the best solution. In nature, termite colonies embody this second approach very successfully. A reasonable compromise may be to omit the expensive centralized controller, have a decentralized system with many cheap robots performing the bulk of construction and a supervisor specifically guiding the last bit of construction.

3.4 Framework Extensions

The current algorithmic framework assumes perfect agents; however, during a long sequence of construction even reliably implemented robots will experience errors such as navigational inconsistencies, failed robots, and improper brick placements. Chapter 4 describes the design and performance of the implemented system and discusses most common failure modes. A major extension to the algorithmic framework, not yet undertaken, will be how to deal with these imperfections to ensure a robust system. On the hardware side this may require more capable robots able to remove unwanted bricks and failed robots from the structure.

Other research directions include the set of admissible structures. Although the algorithmic framework lets structures emerge in different ways, the outcome is always guaranteed to match a user-specified goal and always subject to the restrictions on admissible structures mentioned in section 3.2.4. Building on this framework other algorithms can be developed which greatly expand the class of feasible structures. For instance, section 3.4.1 describes a system where construction can lead to many different structures which share the same qualitative features. Such structures have their likes in both natural and human made construction, e.g. the intricate series of chambers and tunnels in the termite mounds always differ but have the same functional properties, and in suburbias nationwide there is a limited set of possible buildings, but the street layout can vary. Section 3.4.2 describes a scenario where agents are able to both add and remove building material, thus able to build temporary scaffolds to allow for final structures without traversable paths. As mentioned, both extensions still remain loyal to the original approach; the construction is performed by a collective of independent agents who are restricted to the capabilities mentioned in Section 3.2 and utilize local information and stigmergy to coordinate their actions.

3.4.1 Variable Structures

Systems whose outcome depends on the process by which it is built, rather than targeting a specific shape, are more robust to disturbances; e.g. if an obstacle is encountered (such as a rock, or even a failed robot), only part of the structure is damaged and the rest may still grow successfully. Here, I present some simple intuitive examples of how the current algorithm can be extended to work for such systems as well. Please refer to [23] for full details on these algorithms.

Figure 3.4.a shows structures which have been produced with variable outcome algorithms. Both these systems make use of the original agent algorithm and simple agents subject to the restrictions mentioned in section 3.2, with one exception: here, robots must be able to detect the presence of bricks at a distance of up to two brick lengths away.

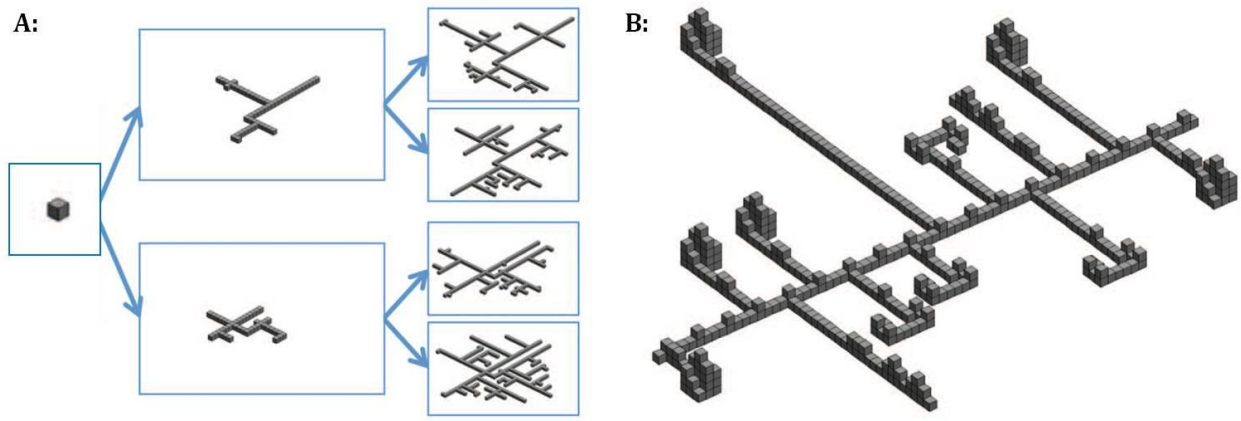


Figure 3.4.a. Results from systems where different sequences lead to different structures because of the local choices of the agents during the course of construction. A: A system producing one-brick high ramifying paths. B: A combination of the original algorithm and that used in (A); here agents stochastically determine whether or not to extend left and right branches and how long they should be, when at the end of a branch they decide what structure to start building and signals this to consecutive agents by specific brick patterns.

To produce the structure shown in Figure 3.4.a.a agents move over the structure and whenever they reach the end of any straight path they decide whether to continue building straight or turn at a 90^{o} angle. Branches terminate when they extend to within a distance of two bricks of other branches to avoid creating narrow tunnels through which two perimeter-following agents cannot pass each other. The key parameter for this system is the probability of starting a new branch at the end of a straight path.

Figure 3.4.a.b shows a more complicated version where branches probabilistically expand from a main path at certain intervals and end in one of three possible structures. The type of structure at the end of the branch is determined by the number of bricks in the second layer of the branch, one brick signals a castle, two a pyramid, and three a linear staircase. If an agent reaches the end of a branch in which second-layer bricks already exists, it must check if the first layer extends more than two bricks beyond those to determine if a structure has already been started. If not, it can choose to modify the choice of end structure or start building it. Using memory, agents can place a brick in the second layer of the main path to signal to other agents that a branch is finished to avoid additional wasted trips. In this type of structure the length of the branches is stochastically chosen during construction as well as what type of structure they end in. However, once agents find themselves on an end structure they adhere to the structpath specific to that structure. As mentioned in the beginning of the section, this system has the benefit that if construction of one branch fails it will not impede global construction.

3.4.2 Temporary Staircases

If agents were able to both place and remove bricks, it would be possible to create structures such as tall towers without staircases. The following algorithm enables construction of any single path structure that allows an entry and an exit staircase next to each other on the exterior boundary. The details of this algorithm are explained further in [25], and in Figure 3.4.b.

Algorithm 3. Agent algorithm for removing a staircase of bricks.

```
1: while bricks left in staircase do
2:     go to structure
3:     follow perimeter counterclockwise until entry point found
4:     climb onto structure
5:     while on staircase do
7:         follow strucpath
8:         if just descended a step
           and not carrying brick then
9:             turn and pick up brick just descended from
10:    discard brick
```

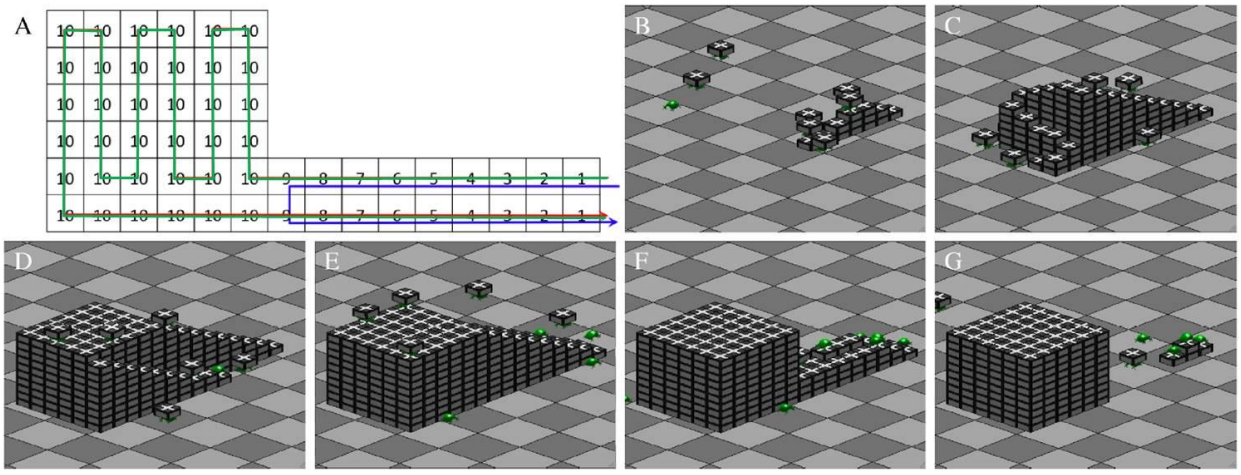


Figure 3.4.b. Process of constructing a tower with temporary staircases. Top: algorithm by which agents remove the bricks in the staircase, following the blue structpath shown in A. A: shows two structpaths, one to build the tower complete with staircase in green and one for the removal process in blue. B-G: shows steps of construction with 10 agents.

The agents build the full structure with a staircase first using the original algorithm, and then as each agent reaches the end of the structure while still carrying a brick it will move off the structure, discard its brick and switch to removal behavior. If an agent enters the structure, but finds an un-climbable wall where the structure should have been, it will switch to removal behavior as well. If an agent on top of the structure is very slow it may end up being trapped as other agents start to remove the staircase, however, with expendable agents that may be okay. Agents remove the staircase layer by layer following a second structpath which only covers the part of the structure that needs to be removed.

Chapter 4.
TERMES: Robotics Implementation

This chapter describes the physical implementation of the TERMES system. I produced a total of 3 robots and 50 bricks and had the system assemble structures many times the size of a single robot. The robots measure 175 x 110 x 195mm and weigh about 800g; each brick measure 215 x 215 x 45mm and weigh 140-210g depending on variations in the manufacturing process. Although the hardware was optimized specifically for the task at hand, the key principles, described in section 4.1, remain valid and could be used in other implementations. Figure 4.a gives an overview of the main challenges. These include locomotion, navigation, manipulation, and how to use multiple robots; sections 4.2-4 describes the mechanical and sensory solution to each, the electronics and control are summarized in section 4.5. Iteration is an important aspect of any successful design, therefore easy fabrication is critical; these methods are described in section 4.6. Finally, section 4.7 evaluates the performance of the system and 4.8 discuss future and exploratory work. The first robotic design was published at Robotics: Science and Systems Conference (RSS) 2011 [24]; the fully integrated system was presented in the Science magazine 2014 [23].

Figure 4.a. Overview of challenges involved in the implementation of TERMES.

I owe great thanks to the many people who helped me design the TERMES system. Dr. Mirko Bordignon from the University of Southern Denmark designed the original software in Python to send high-level commands to the robot from a remote laptop. He also designed the initial routines to make the robot follow the structure grid. Mechanics engineer Rebecca Belisle from Olin College helped design the actuated claw shown in Table 4.4.a. Research staff at the Wyss Institute Christian Ahler helped design the retractable arm described in section 4.4.1 and made considerable improvements to the second version of the ultrasound circuitry shown in Figure 4.6.d. Research staff at the Wyss Institute Dr. Kevin Galloway designed and produced the silicone mold in which the final 50 bricks were cast.

4.1 Approach

As mentioned in Chapter 2 very few algorithms for collective construction have successfully translated into hardware, and even fewer have managed to produce reliable construction. Several design principles contributed to the successful implementation of TERMES: co-design, utilizing passive mechanical features to create a simple and robust system, and reliable control based on error tolerance and recovery rather than a system devoid of errors.

Co-design refers both to the ties between the algorithms and the physical implementation, for example the algorithm requires only functionality which is easy to implement physically, but also refers to the ties between mechanics, sensors and control of the physical robots and the bricks they manipulate - wherever possible complicated sensing and control were diminished by careful mechanical design (Figure 4.1.a). Most issues related to autonomous systems are difficult to predict or understand until they are presented in hardware. With every separate challenge the core of the problem may be identified and solved in a minimalist way. Co-design enables minimalist solutions to every separate challenge, because some are easier solved via the algorithmic framework (Example 1), some by increasing robot abilities in perception and control (Example 2), some by exploiting embodied intelligence (Example 3) and some by exploiting the interplay between structure, bricks and robots (Example 4-5). Construction is one of the few robotic challenges that allow us to shape the environment specifically to simplify the task of the agents, quite similar in idea to how termites re-model their environment to best fit their needs.

Error tolerance and recovery are also critical. Errors are bound to happen in real world systems, especially with construction that need to work over a long sequence of steps. For instance, the TERMES robots work by a number of sub-behaviors, including placing and acquiring bricks, passing over bricks the same height, ascending or descending a brick, turning a quadrant on top of a brick, climbing on to or off the structure and circling it find the seed brick. Hypothetically, if each of these

were made to work with 99% success rate, then performing the 116 actions it requires to build a simple straight-line 8 brick staircase has a 31% (= 0.99^{116}) chance of success. With the addition of multiple robots to the system, the risk of failure becomes even greater. In other words, the key to reliable design is not a system devoid of errors; rather the key is error tolerance and recovery.

Example 1. Intuitively, the TERMES agent algorithm was created to let robots walk along the structpath and upon detection of a missing brick at the next site, deposit one. With the design of the hardware it became apparent that if the robot footprint was to remain smaller than that of a brick (to allow robots to travel over one-brick wide walls), it would be very difficult for the robot to sense anything beyond the brick it was standing on. Rather than complicated sensors, the algorithm was changed so that robots now climb over the site in question before deciding whether or not to add a brick. This was a simple modification in the algorithm, but a great simplification of the robots.

Example 2. To keep robots from colliding, but still allow unsynchronized motion between them, sensors were added to enable perception of other robots up to two brick-lengths away. Control ensures that robots cease activity until a robot ahead of them is no longer visible; short term memory ensures that activity does not commence if the sensors fail to detect another robot intermittently.

Example 3. A single-actuator manipulator enables the robots to pick up, carry and place bricks. Passive actuators based on torsion springs automatically release the brick when the manipulator is in a position to do so, and otherwise keep it in a secure stable grasp without any added control compelxity.

Example 4. The framework specifies that robots must acquire bricks from a brick cache before returning to the structure to find the seed brick. In the algorithmic framework robots magically disappear from sight and reappear with bricks. This issue was solved in the real implementation by connecting the brick cache to the seed, this way robots move only relative to the structure and never have to navigate in open space.

Example 5. The design of robots and bricks are strongly correlated. The choice of robot locomotion (and climbing abilities) define the upper boundaries on brick size. The locomotion chosen enabled robots to climb tall bricks with simple control; however, it also caused a wobbly unstable gait. Rather than complicating the control of the robot to achieve accurate locomotion on top of structures, physical features in the bricks passively align the robot as it tries to turn and climb. Furthermore, bricks have self-aligning features which make them snap easily into place when added to the structure

Example 6. Despite careful design, robots do not always manage to climb bricks in their first attempt. Rather than complicating the mechanics for locomotion or decreasing the height of the bricks, inputs from an accelerometer helps the robot know if it has succeeded or if it should continue trying.

Figure 4.1.a. Examples to stress the need for co-design and error tolerance.

The robots must have sufficient sensory capabilities to detect if errors happen within each behavior and correct them before they become fatal to the entire system (Example 6). Ideally, error recovery should be implemented at the algorithmic level as well; however, we have yet to explore this option.

Figure 4.1.b shows the process through which the TERMES robot hardware and control was designed. The overall process is quite general and many aspects will remain the same for other implementations of collective construction. In the following text I describe how the problem naturally breaks up into several sub-challenges, and some guidelines for how these challenges can be effectively addressed. As mentioned earlier, I believe several design principles are important for any physical implementation. I favored simple designs over complex actuation, sensing and control, I exploited co-design wherever possible, placing as much control as possible into passive mechanical features to make it physically hard for robots to commit mistakes, and I designed feedback systems to allow robots to recover from errors. Finally, I placed emphasis on easy and fast fabrication to enable quick re-iteration of the mechanics.

The physical implementation for collective construction can be broken into several sub-challenges:

Locomotion. The first challenge is to design a simple mechanism for locomotion on and off built structures. Good climbing abilities enable tall bricks, and the taller a brick the robot can scale, the more material can be added to the structure with every deposition. Therefore, better climbing abilities lead to faster completion of structures. However, an unstable gait complicates navigation on top of narrow structures. Because robots must be able to climb with and without a brick, both weight distribution and ground clearance are important factors in design. The TERMES robots use simple differential steering along with special type of wheels to achieve good climbing abilities and relatively stable gait on level ground.

Navigation. The second challenge is that of navigation on and off the structure. Navigation on the structure needs to be accurate to keep the robot on the structure grid; the tolerances of these maneuvers place limits on the width and length of the bricks. The TERMES system solves this challenge by passive mechanical features on the bricks that help guide robots along the structure, avoiding the need for complicated sensing and control. Navigation off the structure can be simplified by having the robots navigate relative to the structure only, for example I exploit wall-following behavior which is known to be robust and flexible. Finally, in TERMES, the brick cache is positioned next to the seed brick so that robots never have to leave the structure to obtain building material and never have to localize in unrestricted spaces. Limiting navigation to use local sensing has several advantages, local sensing is easier to implement and make reliable, compared to long-distance ranging or global position sensing.

Manipulation. The third challenge is to reliably pick up, carry, and place bricks. Designing good manipulators are a complex problem; most off-the-shelf robot grippers are not well matched to their problem because they are optimized to grasp many types of objects and have low precision. It has been shown, even for general grippers, that well-designed simple mechanical features can outperform grippers with complex actuation and control [73]. The TERMES robots are required to manipulate only a single type of objects, but must do so accurately; this is achieved through co-design of the gripper and material. The TERMES robots use just a single-actuator manipulator which automatically keeps bricks securely in place while effortless releasing them when desired. The bricks include a handle to aid reliable grasping and physical shapes and magnets which help bricks snap together despite robot misalignments.

Coordination. Finally, with a single robot incorporating all of these features robustly, collective construction with multiple robots can commence. Coordination can be a complex thing to implement, especially as the number of robots gets larger. Planning-based approaches where each robot is given a different building plan and must control both location and timing relative to other

robots, imply a large cost in hardware development. Here, decentralized and minimal coordination at the algorithmic level substantially simplifies the requirements of the robot hardware; robots are only required to perform collision avoidance with respect to other robots. I implemented this by having robots listen for and emitting a warning signal to their immediate vicinity.

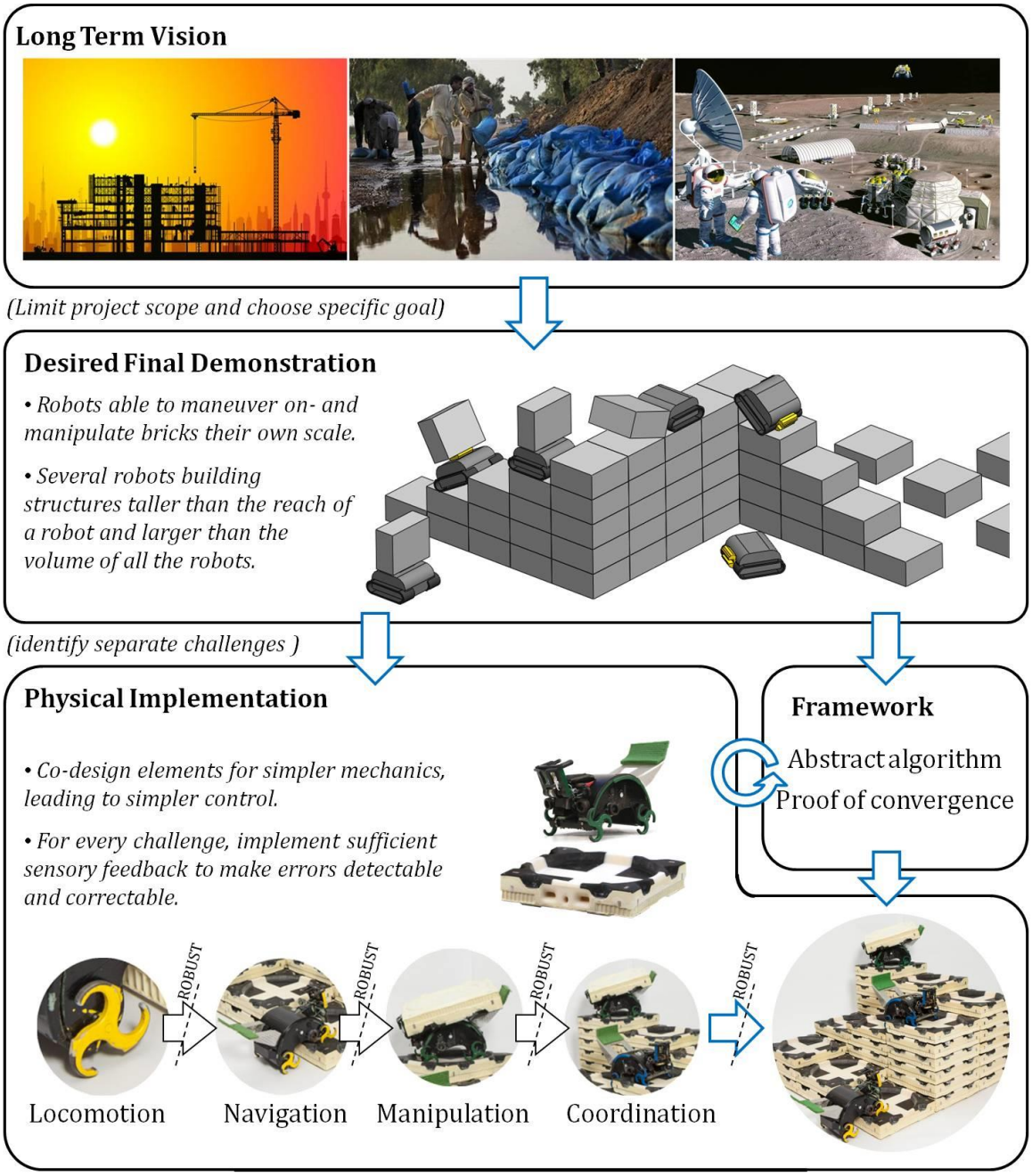


Figure 4.1.b. Suggested process for designing a successful system for collective construction.

The result of this process is a highly optimized system where the robots work only as an extension of the bricks for which they are designed. The advantage is a reliable, yet simple solution to a complicated problem. However, this system is still restricted to a lab environment; I believe that full-scale solutions able to construct structures in real-world scenarios will only emerge from the lessons learned through many iterative cycles of bio-inspiration and robotic systems, of which the work presented here is one.

4.2 Locomotion

The TERMES system consists of independent robots constructing on scales much larger than themselves. This poses the challenge of designing reliable locomotion on both flat and angled terrain. This section presents the design of a robot able to locomote on level ground and on structures, as well as climb up and down the height of a single brick.

The field of climbing robots is vast. Some designs target vertical walls by means of gecko and spider inspired adhesion [74, 75], electroadhesion [76], suction cups [77], magnetic treads [78] and some even climb trees by clamping onto the trunk [79]. Many designs exist that are able to climb over more restricted inclines, for example climbing rugged terrain and stairs with legged robots [80, 81], wheels [82], and combinations of wheels and legs [83, 84]. As will become apparent throughout this section, the TERMES robots are similar in spirit to the latter; they use a combination of wheels and legs, termed "whegs" [85] that are especially useful for easy control over unstructured terrain. However, the design of TERMES is further optimized for stable and accurate locomotion when on top of structures. Here, I describe the locomotion design that allows simple robots to reliably climb and maneuver on tall slim bricks, by exploiting passive mechanical features, rather than added control complexity.

4.2.1 Design Process

To speed up construction, the volume of material depositions must be maximized. Deposition size is determined by how much a robot can manipulate and, even more so, by how much it can climb and maneuver on (Figure 4.2.a.a). It is important to remember that this implementation is a proof-of-concept; for the framework the exact size of the system is irrelevant, be it on the scale of a termite, a standard brick, or a cement truck. The *proportions of the system* are important, however.

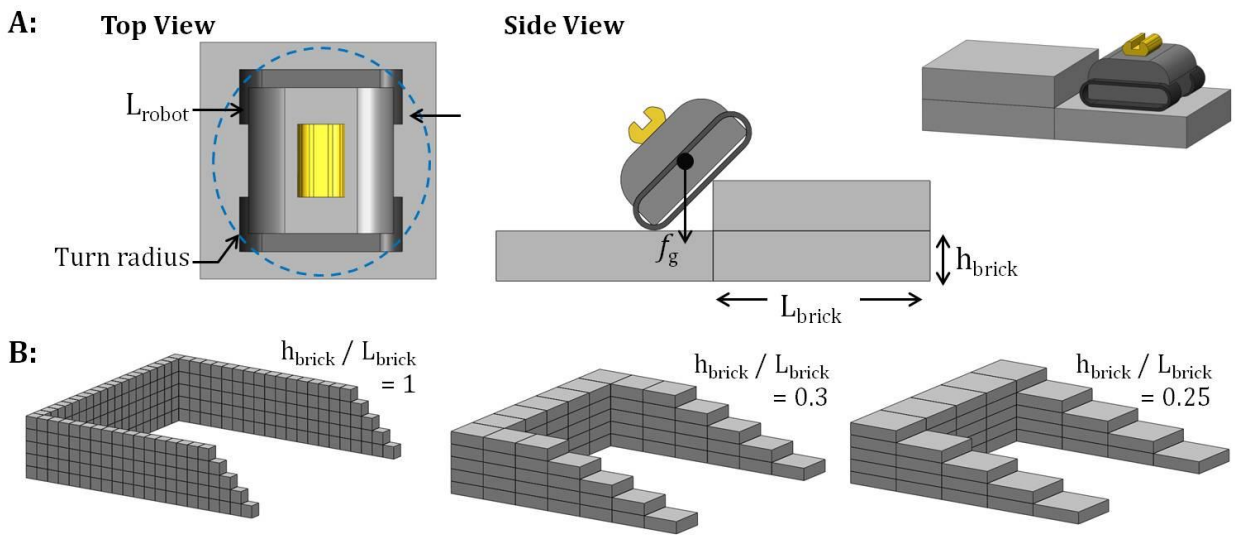


Figure 4.2.a. A: Illustrations of parameters which affect robot climbing abilities, as well as brick size. The side view shows how robots must be long enough to keep the center of mass from tipping the robot backwards. It is assumed that the weight of the brick is small compared to that of the robot, so that a robot carrying a brick will have approximately the same center of mass (discussed further in section 4.4). B: Three solutions to the same goal structure; dependent on brick height-to-length ratio. The structure to the far right is optimal in resolution and embedded volume.

Specifically, the goal is to optimize the height and length of the bricks, h_{brick}/L_{brick}, i.e. to design robots able to climb bricks as tall as possible, while still being able to maneuver on a brick surface as small as possible. Brick length is influenced by several factors. For example, for a desired shape, decreasing the length of the brick increases the resolution of the structure, which can be desirable (Figure 4.2.a.b). A lower bound on brick length is determined by the robot's ability to maneuver reliably on top of a single brick. Brick height h_{brick} is constrained by climbing ability. In previous work I designed robots to climb square bricks their own height [31]; they did this using arms and claws dedicated to the purpose of climbing. This required complicated and fragile mechanical parts, strong actuators, accurate sensing, and detailed control resulting in a very unreliable system. Building on the lessons from that project, the locomotion of the TERMES robots is based on much

simpler mechanics and just two actuators, with the trade-off of lowering the climbing- and thereby the brick height.

I designed a reconfigurable robot base (Figure 4.2.b) to compare different strategies for climbing and locomotion on level ground. All 15 configurations tested used only two actuators, differential steering, simple forward propulsion for control, up to 90g weights for balance, and a combination of wheels, whegs, and treads. For simplicity, robustness, and reliability I avoided complex designs dependent on more than two actuators, like ones that use flippers [86], for climbing. The robot was self-contained with on-board power and control based on an ATmega1281 and code written in C.

Each robot configuration was compared with respect to three parameters:

- *Maximum climbing height* with an allowed failure rate of 20% (timed out after 30s).

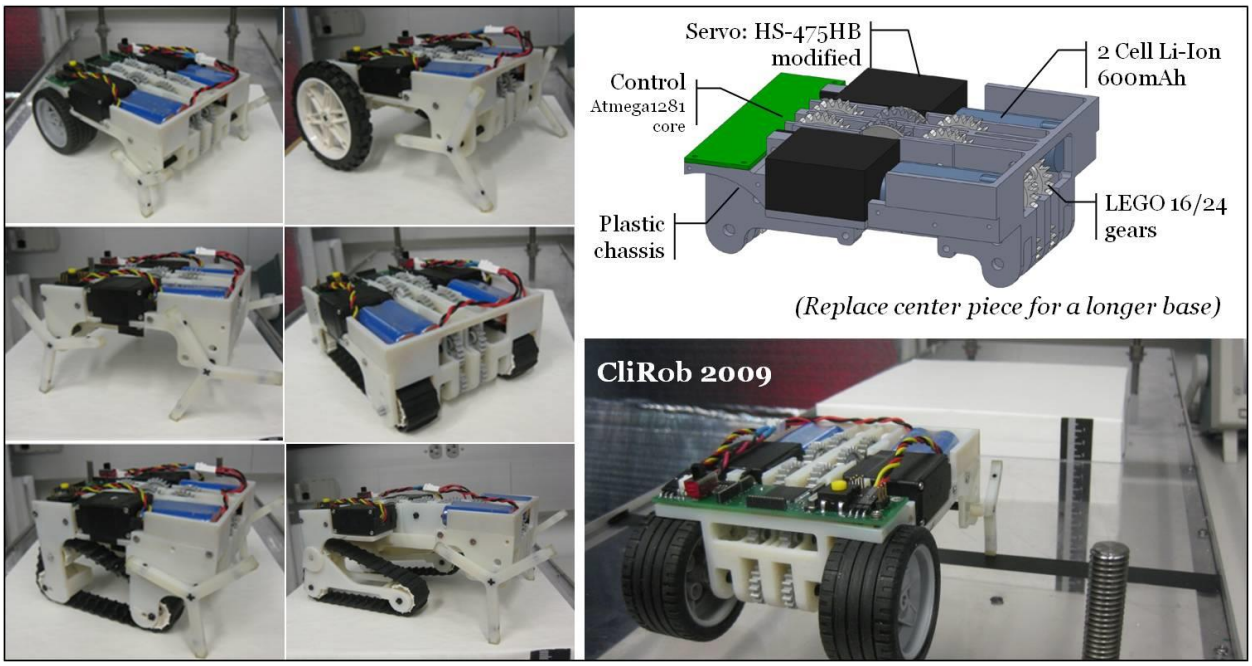


Figure 4.2.b. Reconfigurable climbing robot (CliRob). Left: 6 out of 15 configurations. Upper right: Model of the reconfigurable robot base. Lower right: Test setup with adjustable brick height.

- *Ability to automatically align* as the robot is climbing the structure. This is a good example of embodied intelligence; many configurations keep pressing up against the brick until they are well aligned before they are able to successfully scale it.
- *Gait smoothness* on level ground. Smooth gait simplifies the design of sensors and manipulators in the final robot.

The results can be seen in Figure 4.2.c. As the figure indicates, the final design was based on all whegs, trading off gait smoothness for climbing and automatic alignment abilities.

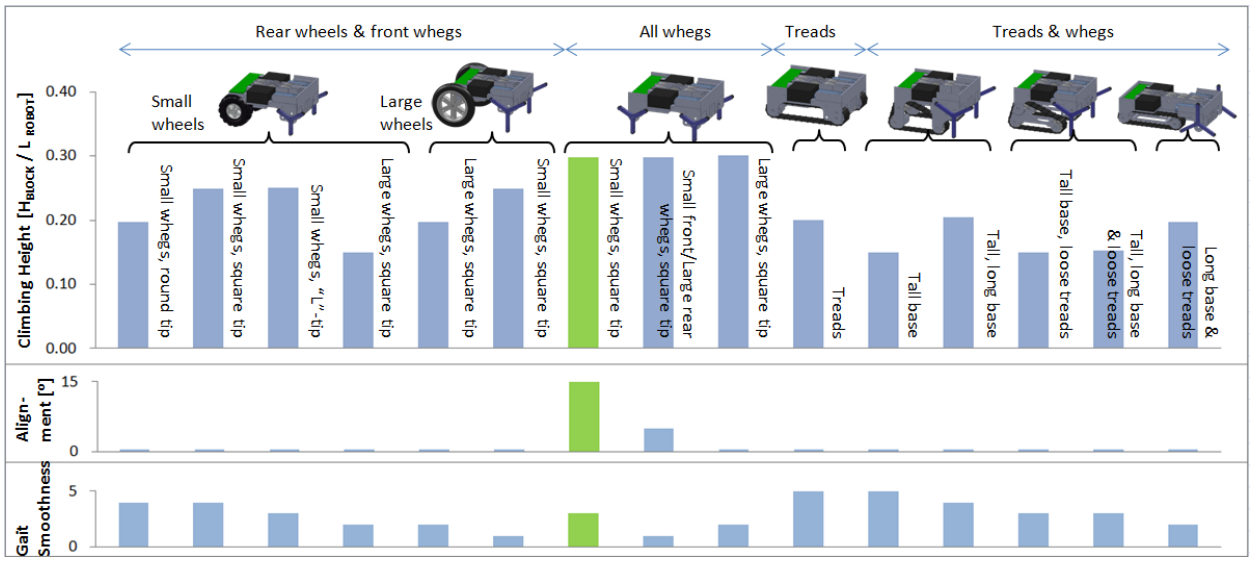


Figure 4.2.c. Top bars show the height of the tallest brick the robot was able to climb successfully in 8 out of 10 trials. Middle bars show the maximum angle, Θ, for which, if the robot approached the block at Θ from perpendicular, it would straighten itself out in the process of climbing. Lower bars give a qualitative rating of gait smoothness on a level surface (5=smoothest travel, 1=most lurching). Base configurations use short length and low height unless otherwise noted.

4.2.2 Final Design

The final robot locomotion system is based on differential steering and whegs, requiring just two actuators and simple control. The robot easily travels over level ground, climbs bricks with a height to length ratio of 0.36 (= 45mm/215mm), and uses mechanical features on the bricks to help perform accurate maneuvers while on the structure.

Figure 4.2.d shows the final robot design. In addition to the choice of whegs, I also adapted the body shape of chassis to have a raised bottom to clear the corner of the brick as the robot climbs (4.2.d.b). Figure 4.2.d.a shows the details of the robot drive train. Two metal gear motors mounts in the rear of the robot and are held in place by a shelf which also supports a battery pack. The motors are geared up slightly before connecting to the front and rear axis via timing belts and pulleys. The continuous top speed of the robot is approximately 13cm/s.

Figure 4.2.d.d highlights the many passive mechanical features on the bricks which help the robot climb and perform accurate maneuvers on top of the structure:

- Notches on the side of the brick help the robot climb even taller bricks. The notches are chamfered to promote automatic alignment if the robot has drifted off center; both while climbing and while travelling over bricks the same height. The bottom of each notch is covered in caulk to add traction for the robot whegs while climbing.
- A bowl indentation helps passively restrict the turning radius of a robot on the structure; the robot will only be able to climb out of the bowl if it actively tries to do so.
- Filleted and chamfered edges around the rim of the bowl and the inner corners of the brick prevent the whegs from getting stuck and ease the casting process of the bricks.
- Inverted features matching those in the top are added to the bottom of the brick to make them passively align and stack almost like LEGOs.

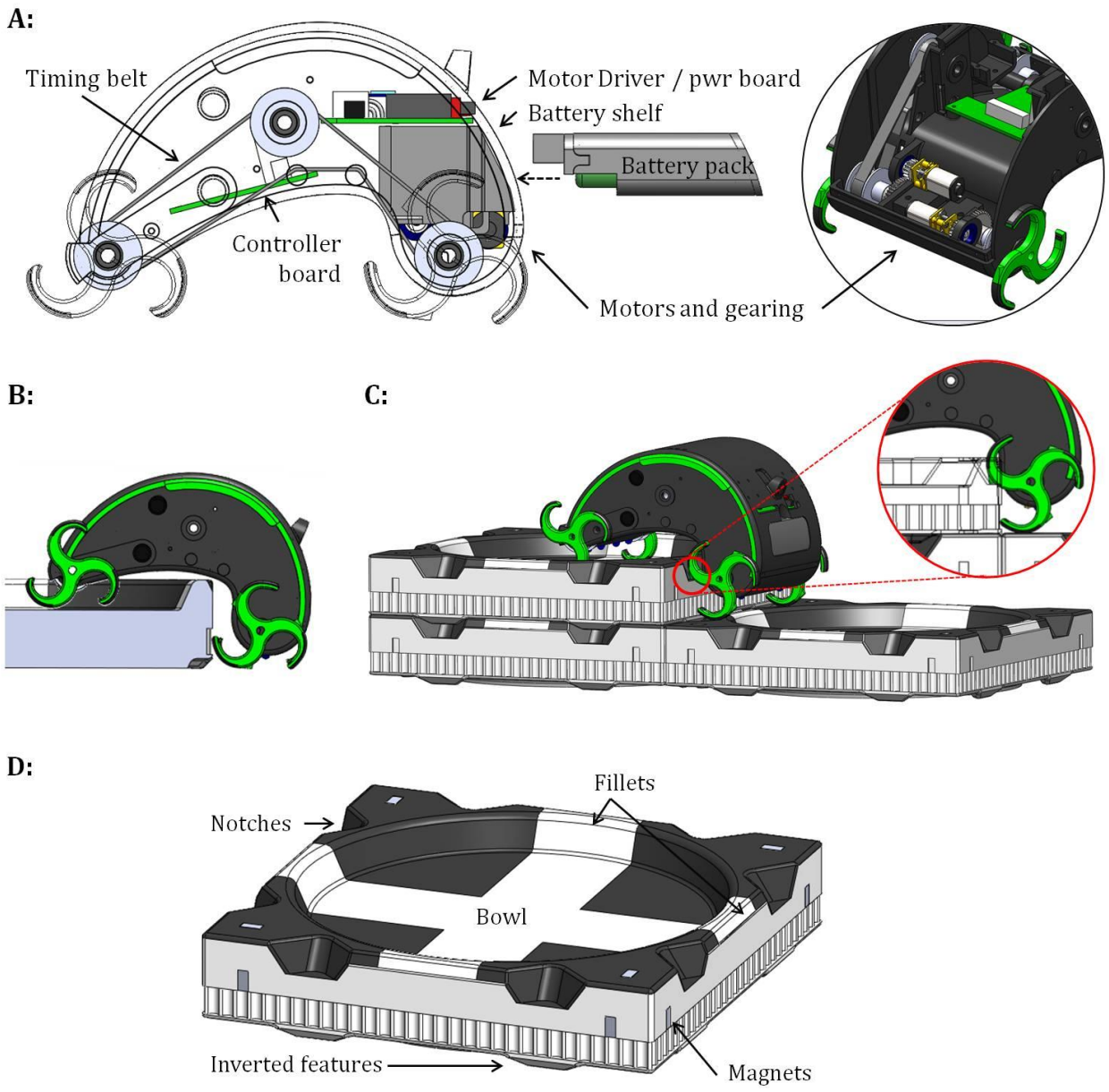


Figure 4.2.d. A: Cross section of the robot showing the drive system, controller and power electronics, the battery pack and the shelf it rests on. B: The raised center of the robot chassis helps it clear the corner of a brick as it climbs from the ground. C: Robot climbing up bricks, using hooks on the rear whegs for added traction. D: Brick showing features related to climbing and attachment.

- Magnets are added on all faces of the brick for structure stability and alignment[1].

Similarly the shape of the whegs represents several detailed design choices to help the robot climb and maneuver (Figure 4.2.d.c):

- The wheg orientation is deliberately chosen. Had they been flipped to work as hooks it would have increased the risk of getting stuck while climbing (between the wall in front and the notches of the brick below it). In this orientation, the primary function of the whegs is to enable climbing of steps as tall as their radius.
- Increasing wheg radius increases the possible climbing height, but with the cost of a more wobbly gait and less stability while climbing because of the raised center of mass. The current radius, 28mm, was based on a mix of empirical studies and model based optimization.
- The curved legs decrease climbing height, but highly improve turning radius. It also helps to make the gait less wobbly, easing the process of sensor calibration (details in section 4.3.1).
- The outer curve of each wheg is covered in rubber to improve traction while climbing; the tip is hard plastic to avoid it catching on the rim of the brick when the robot turns in place.
- Hooks on the rear whegs fit in the bottom of the brick notches and increase traction while climbing.

[1] The current magnets are relatively weak (Neodymium magnets with a holding force of 1.56lbs when pulled from a steel plate). Adding stronger magnets will make brick attachment easier, however, these specific ones were chosen to make the current robot compatible with a possible future system of smart bricks exploiting magswitches of the same strength, see section 4.8.3.

4.3 Navigation

Robots must be able to autonomously navigate on and off the structure. On the structure a robot must follow the structpath and detect missing bricks; off the structure a robot must find its way back to the seed brick and pick up new material on the way. The problem is restricted to a structured laboratory setting, with level black floors of uniform color, steady lights, and low noise.

As mentioned in section 2.3 most hardware demonstrations in collective construction exploit global perception [9, 39, 41, 42, 48, 50, 53]; of the ones that rely only on on-board sensors [31, 49, 51, 55] few have made it beyond two dimensions and none have shown reliable performance in three. Here, I present robots restricted to on-board sensors reliably navigating 3D structures, using only infrared light (IR), ultrasound, and a 1-axis accelerometer.

4.3.1 Design Process

This section is divided into two separate challenges, that of navigation on- and off the structure.

4.3.1.1 Navigation on the Structure

When operating on top of a structure the robot has to keep track of its location in the structpath and the configuration of the last two bricks it has passed, see section 3.2.2. It is important to remember that although the structure provides a grid on which robots can navigate, they are not physically restricted to stay on that grid. Here, I describe how robots accurately and reliably perform navigation on top of bricks with small footprints[1].

[1] As described in Section 4.2, for fast construction progress, the height-to-length ratio of the bricks should be as big as possible; in other words the footprint of the brick should be as small as possible, while still allowing reliable navigation.

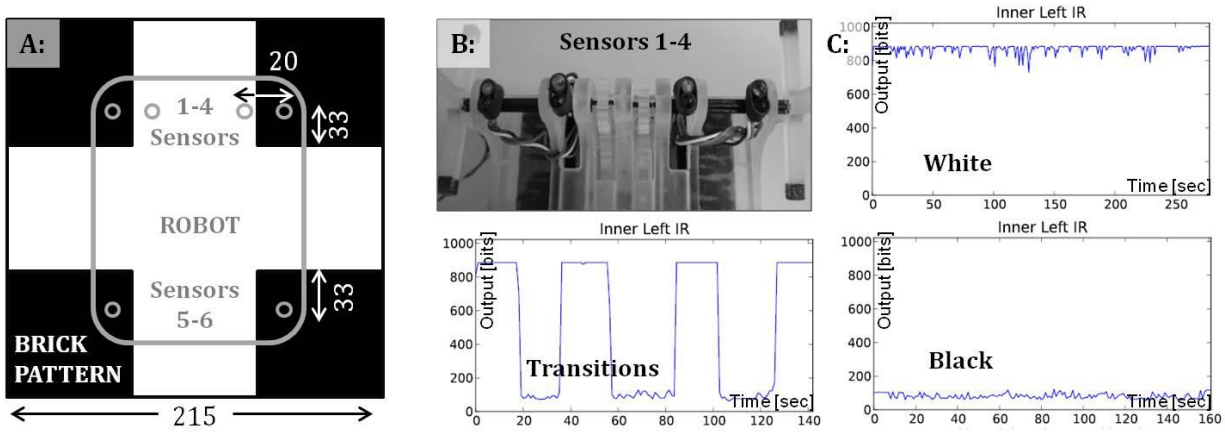


Figure 4.3.a. Brick patterns and IR transceivers. A: Dimensions of grid pattern and position of sensors, top view, all dimensions are in mm. B: Configuration of transceivers in CliRob. C: Example of output from a sensor as the robot drives over black and white surfaces.

To allow the robot to perceive the structure grid, IR transceivers were added to the belly of the robot and the top of each brick was shaded in symmetric cross patterns so that robots could enter from any perpendicular angle. Four IR transceivers mounted in the front and two in the rear provided sufficient sensory feedback to move straight between bricks. The width of the pattern and the sensors on the robot were positioned according to two requirements. First, a robot must be able to detect when it is positioned over the center of a brick fairly accurately (Figure 4.3.a.a-b). Second, reversing until the front sensors all detect white must displace the robot on the brick enough to operate the manipulator unhindered (see section 4.4.2 for details), but not so far that it risks falling backwards off the structure. The IR light was modulated to restrict ambient light influence, and configured so that black and white colors saturated the sensors independent of the wheg position and distance of the sensors to the surface underneath (Figure 4.3.a.c).

The biggest challenge was to make the robot able to turn in place. Initially, the whegs were composed of four thin spokes, with a wheel radius of 20-29mm, causing severe drift when turning open loop. To keep the robot in place while turning, software was devised to continuously correct

the position with respect to brick patterns in three shades of grey (Figure 4.3.b.a). Several iterations of the whegs eventually resulted in those seen in Figure 4.3.b.b with a softer shift in radius over a rotation cycle (radius 22-28mm). Despite these improvements, control to enable reliable turns became very complex. To solve this problem I implemented a physical indentation in the bricks; now, unless the robot is actively trying to climb out, an indented bowl will keep it in place while turning (Figure 4.3.b.c).

In addition to moving forward and turning, the robot also must keep track of its height position on the structure and know when it has ascended or descended a brick. For this purpose, I implemented a 1-axis accelerometer; as the robot passes from one brick to the next it continuously monitors the output of the accelerometer and thereby determines if it is moving over bricks the same level, or climbing up or down a level. When the robot detects a pattern indicating that it has moved from one brick to the next, it checks the accelerometer to ensure that it is on level ground; if it slipped while climbing it could have ended up halfway between two bricks.

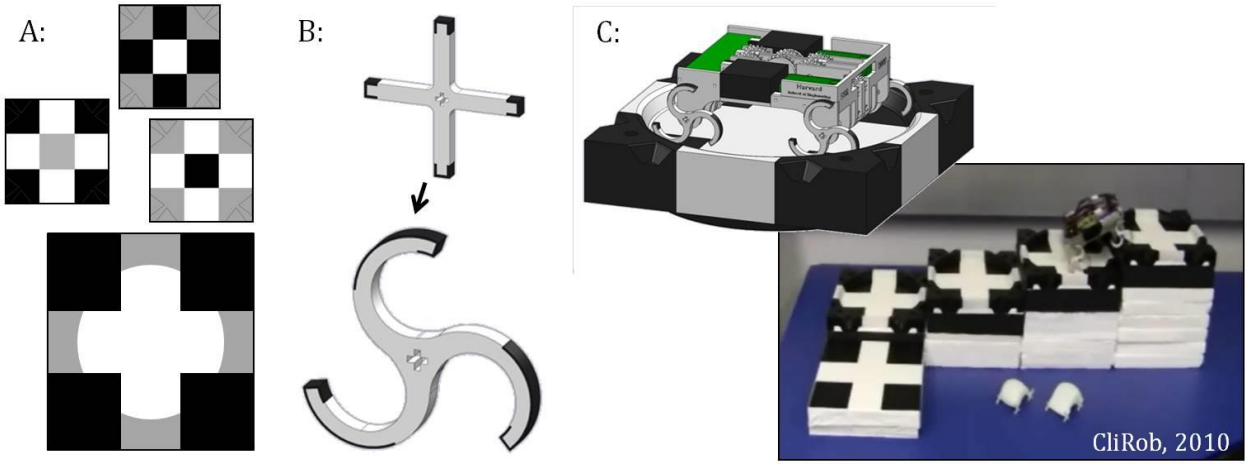


Figure 4.3.b. Co-design of robots and bricks to solve the challenge of autonomous navigation. A: Initial top patterns on the brick, to guide turning. B: Change from spokes to curved whegs to decrease the turn radius. C: Final black and white pattern and bowl indentation.

Finally, the robot uses an 'edge sensor' to detect if the next step in the structpath is on the structure or if the structure has come to an end. The edge sensor is simply an extra IR transceiver placed in the tip of the manipulator, so that robots can move to the edge of the brick on which they are standing and check for cliffs ahead. The edge sensor is important for two reasons. 1) If the robot has lost track of its position in the structpath and come upon an un-climbable drop in elevation, it must detect so and try to find its way off the structure unharmed, see section 4.8.4 for details on adaptive robot behaviors. 2) To add a brick on the ground level, robots must detect when they come to the end of an unfinished structure, turn around and reverse straight off of the structure before placing the brick. More intuitively a robot would turn after it has climbed off the structure; however, because of the large drift in physically unrestricted turns that approach is not possible.

In general, although a 4-whegs configuration works well for locomotion in unstructured terrains, I found that their use complicates accurate navigation. Traction changes dependent on which parts of the whegs are in contact with the ground, hence the poor turning-radius, and sensor treatment must be tolerant to the fact that their position changes relative to the environment as the whegs rotate; sometimes in synchrony, other times off phase with each other.

4.3.1.2 Navigation off the Structure

Navigation off the structure involves navigating from a position at the end of the structure back to the seed brick and picking up new material on the way. To simplify the implementation, I placed the cache of new building material next to the seed brick, and had robots follow the perimeter of the structure until they found the seed brick and could re-enter the structure (Figure 4.3.c.a). The seed brick itself was marked by a white line on the black floor, which was easily detected by the downward facing pattern sensors on the robot.

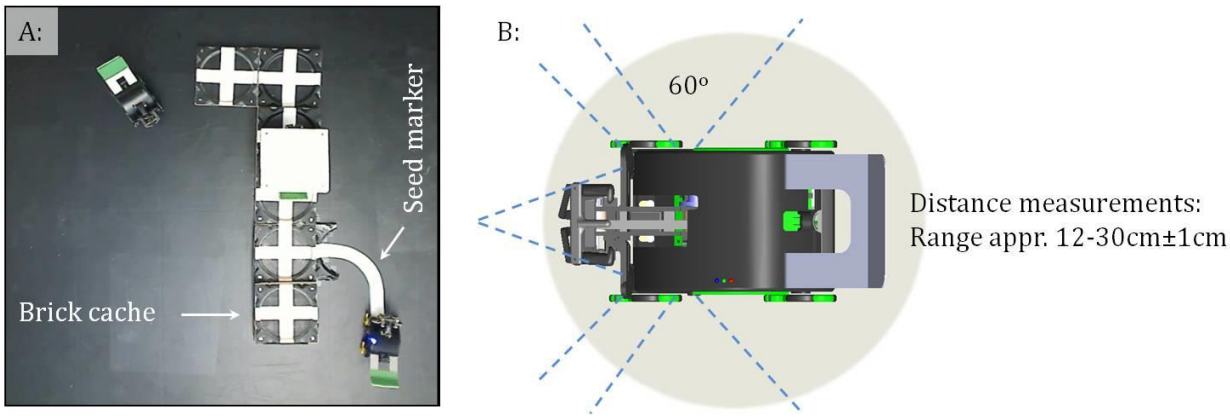


Figure 4.3.c. Features implemented in the structure and on the robot to help it navigate off the structure. A: Top view of a structure, with a white line marking the seed brick, and the brick cache with new material located next it. B: Top view of a robot showing four ultrasound transceivers for ranging.

I implemented a simple and robust wall-following approach that allowed robots to follow the structure to the seed marker. For structures with enclosed spaces, the structpath restricts exit points so that robots do not get trapped in any internal perimeters. Wall following was accomplished with ultrasound distance sensors to measure distance from the structure. I chose ultrasound so that it would not conflict with the IR robots use to navigate on the structure. Emitters and receivers were mounted, two in the front and one on either side of the robot. Robots always follow the perimeter strictly counter clockwise. Having sensors on both sides of the robot allows it to detect tunnel entrances which are too small to allow two robots to pass each other and consequently should not be entered. The two sensors in the front further help align the robot correctly when trying to expand the structure on the ground plane.

Several sensors and configurations were tested; the final choice was picked mainly for its short ring-times, allowing measurements as close as 12cm. The position and angles of the transducer match an optimal distance of 15cm from the structure. To avoid direct coupling between transmitter and receiver, as well as reflections from the floor overpowering reflections from nearby

objects, rings were molded in soft absorbing foam to fit around the transmitter. A routine to perform automatic sensor calibration was implemented to help easily accommodate changes to the robot chassis.

4.3.2 Final Design

The robot performs autonomous navigation relative to the structure using just four types of simple sensors and passive mechanical features in the bricks (Figure 4.3.e).

The robots are able to-recognize and follow the structure lattice using:

1. Black and white symmetrical patterns on the surface of the bricks.
2. Six IR transceivers underneath the robot for pattern recognition.
3. Indentations in the surface of the bricks to help alignment and turning in place.
4. An edge sensor to detect the end of the structure (and cliffs in case of errors).
5. A 1-axis accelerometer to detect whether the robot is passing straight between two bricks, ascending or descending a step.

The robots are able to navigate off the structure using:

6. An ultrasound transceiver on either side to perform ranging, as they follows the perimeter of the structure back to the seed brick.
7. Two ultrasound transceivers in the front to avoid frontal collisions and to align themselves with the structure before adding bricks on the ground.
8. Rippled edges on the side of the brick, to return an emitted signal independent of the angle between the robots and the structure.

9. A white line on an otherwise black floor indicating the seed brick, to be detected by the IR pattern sensors underneath the robots (Figure 4.3.c.a).
10. A brick cache next to the seed brick, so that they never have to leave the structure to find new material (Figure 4.3.c.a).

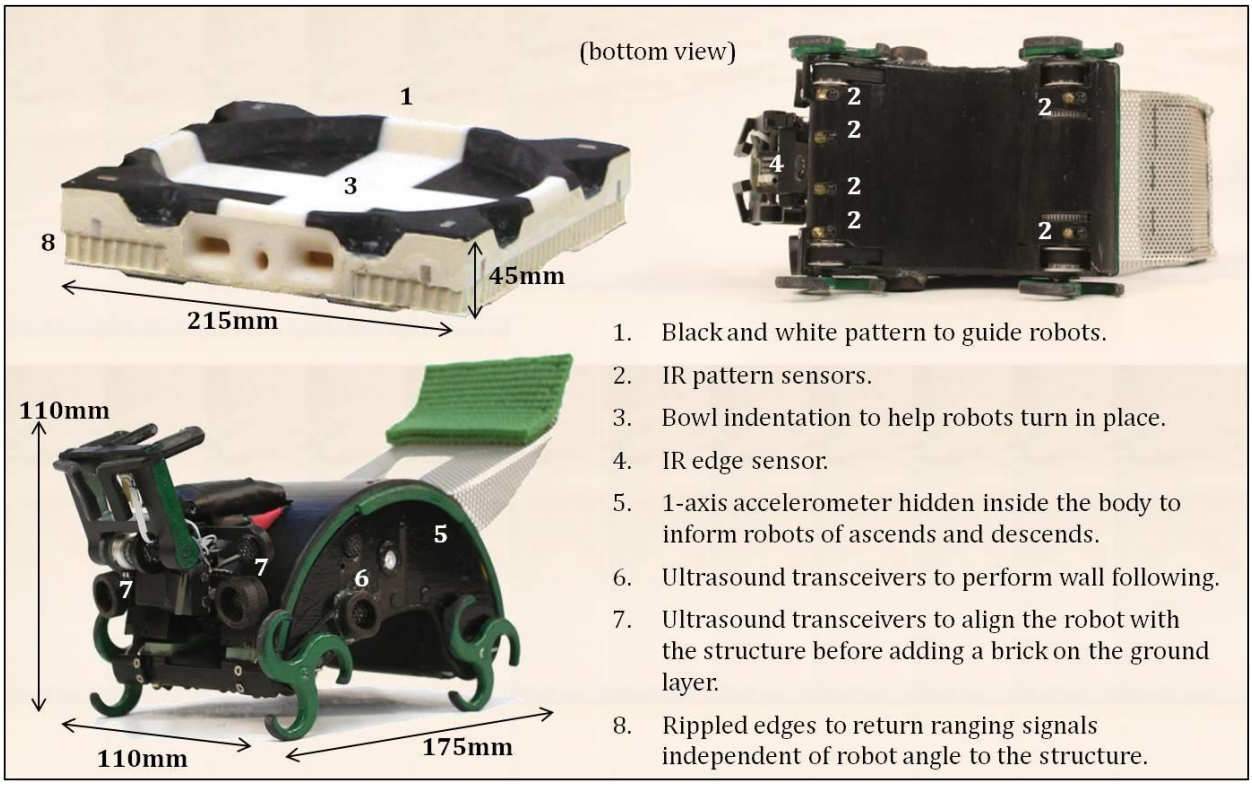


Figure 4.3.e. Features and sensors on robots and bricks enabling multiple robots to perform autonomous navigation reliably on and off the structure.

4.4 Manipulation

Robotic manipulation is a vast research area with challenges in mechanical design, tactile sensing, modeling and control [87]. Construction robots only need to be able to manipulate a limited set of objects in a limited manner (pick up, transport, and place bricks), but must do so with high accuracy. These requirements limits the design space; in the spirit of minimalism and embodied intelligence I addressed the manipulation challenge by a) co-designing claw and handle, shaping the tool for the hand and the hand for the tool, and b) limiting the number of DOF required.

The following sections describe methods for brick transport, early and final versions of the manipulator including arm, claw and handle, and how the center of mass is positioned to let the robot climb with and without a brick.

4.4.1 Design Process

Insects often handle objects the size of their own body and they transport these in widely different ways, see Figure 4.4.a. Some push or drag the object; long-legged army ants hold their load underneath their body, leaf cutter ants hold their pieces high in the air, and others, like many species of termites, swallow the load and later use their feces to construct tunnels.

Figure 4.4.a. Insects manipulate material in widely different manners; by pushing or pulling, or by carrying it underneath, over, or even inside, their own body.

I considered many options inspired by this diversity. Storing a brick inside the robot body has the benefit of a fixed center of mass independent of load. Unfortunately, this approach increases robot size beyond the footprint of the brick, and therefore its ability to travel near a tall wall. Storing bricks underneath the robot will obscure the view of the pattern sensors. Hence, the robot must transport bricks on top of its chassis. Transporting bricks by holding them vertically, such that the slimmest side is facing forwards, would allow the robot to pass down narrow tunnels. However, I chose to keep the brick lower and flat against the top of the chassis to 1) obtain a more stable grasp, 2) lower the center of mass for easier climbing, and 3) limit the DOF needed in the manipulator.

4.4.1.1 Robot Manipulator

The robot manipulator must be able to do five tasks: grasp a brick placed at its own level, lift the brick onto its chassis, carry it securely while climbing up or down, bring the brick and claw back down, and place the carried brick in front of itself at the same level. Several manipulator designs were tested with respect to reliability, robustness, need for sophisticated control, electronics and power, as well as the size and weight of the claw and the corresponding handle in the brick. Each configuration was attached to an arm which could rotate the claw, with and without a brick, and on and off the back of the robot. The arm itself was mounted on a robot (Figure 4.4.b), and software was devised to autonomously control the robot to grasp (attach to a brick and raise the claw), maneuver (turn in place, ascend and descend a two-brick-tall structure), and finally place a brick (lower the claw and detach). The brick itself was cut out of Styrofoam, and a metal ball was added to the center to imitate the estimated weight of a final brick (~172g). To ease the design of the manipulator a special brick without magnets on the top surface was designated as a docking station from which robots pick up new material while standing on an adjoining brick. The docking station

is manually reloaded every time a brick is removed. Docking was complicated by the fact that the manipulator had to tolerate a 12mm height variation caused by the whegs. Table 4.4.a shows the evaluated designs, and how elastic grasping was chosen. The four designs include:

- A passive two-pronged claw. A simple and mechanically robust design, with a filleted handle to ease docking. The depth of the handle, and therefore the length of the prongs, was limited because of the indentations in the brick. Short prongs caused instability and despite optimizations the brick often fell off, either during the rotational movement onto the robot back, during transport, or prematurely during the rotational movement off the robot.
- An elastic three-pronged claw. A fairly robust design in which the two outer prongs were elastically pulled towards each other, and a third fixed prong added in the center for stability. When the claw was lowered against a wall in the robot body, the two loose prongs were forced outwards and easily grasped on to a filleted handle in the brick. When the claw was raised the prongs automatically snapped into internal divots in the handle. This feature drastically improved the hold on the brick during transport.

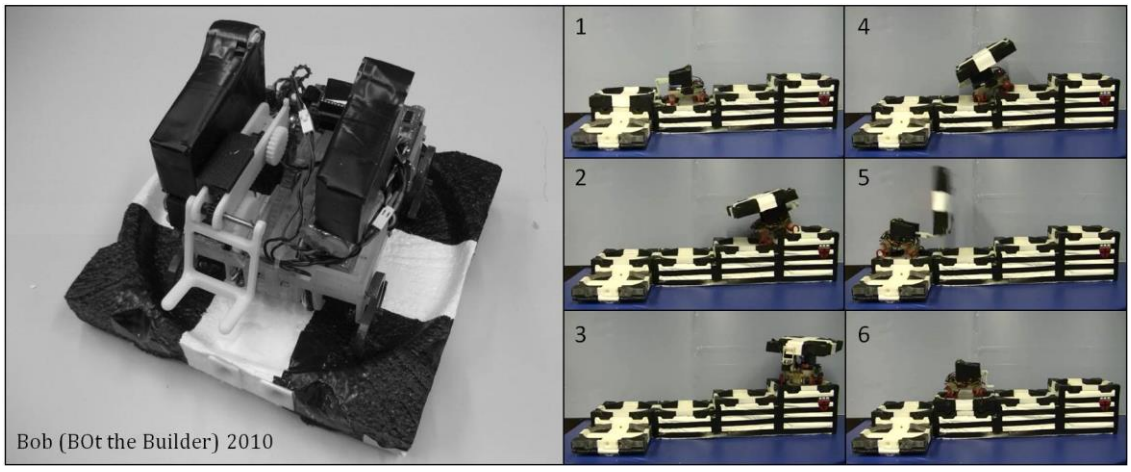
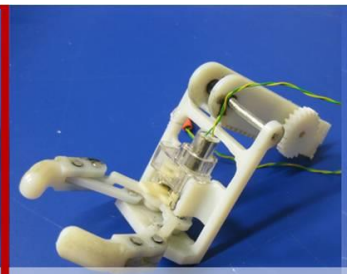
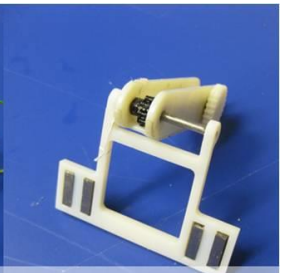


Figure 4.4.b. Left: The robot, Bob, with an arm and a passive claw, sitting on top of a handcrafted brick. Right: A sequence of snapshots from a demo concerning brick manipulation.

- A claw with two actuated prongs for stability during transport. When off, a torsion spring in the actuator forced the prongs to a tight grasp; when on, the motor opened the prongs wide. This design was very reliable in all tasks, but did have a lot of fragile parts, added weight, and the need for well timed control.
- A magnetic claw. This was a mechanically simple design with electro-permanent magnets in the claw latching to steel bars in the brick to eliminate the need for handles and good prior alignment. The main limitation was the need for a very strong magnetic coupling to overcome the shear forces while lifting or placing the brick. Unless combined with mechanical features this approach was not reliable.

Table 4.4.a. Evaluation and results from performance tests of four different types of claws.

 Passive Grasping		 Elastic Grasping	 Actuated Grasping	 Magnetic Grasping
Robustness	High	Moderate	Low	High
Power	-	-	X	X
Claw Weight	Light	Light	Heavy	Heavy
Claw Size	Moderate	Moderate	Large	Small
Electronics	-	-	X	X
Timing/ Control	None	None	Much	Some
Brick intf.	Handle	Handle	Handle	2 steel bars
Success rate				
Grasping	9/10	9/10	10/10	6/10
Holding	5/10	8/10	10/10	9/10
Placing	8/10	9/10 **1. Choice**	9/10	7/10

4.4.1.2 Robot Arm

The robot arm also poses an interesting challenge. To grasp a new brick from the docking station while standing on an adjoining brick, the claw of the robot must reach beyond the footprint of the brick it is standing on. However, during normal maneuvering it is better for the robot footprint to remain smaller than that of a brick, e.g. so that turning next to a wall is physically possible. Therefore, the initial design of the arm was made to retract into the robot body after docking.

Many versions of this retractable arm were made, and the final prototype version is shown to the left in Figure 4.4.c. This arm used a single actuator that combined linear retraction of the arm with rotary movement of the claw without the need for additional control. However, after several months of careful design the largest structure assembled without errors was still only two bricks wide. The complex design of the arm was too prone to error to achieve the reliability needed for large scale construction. Instead, the final design of the manipulator uses a fixed arm, which permanently protrudes from the body of the robot. The trade-off is that the robot must move to the very back of the brick underneath to lower the claw unhindered.

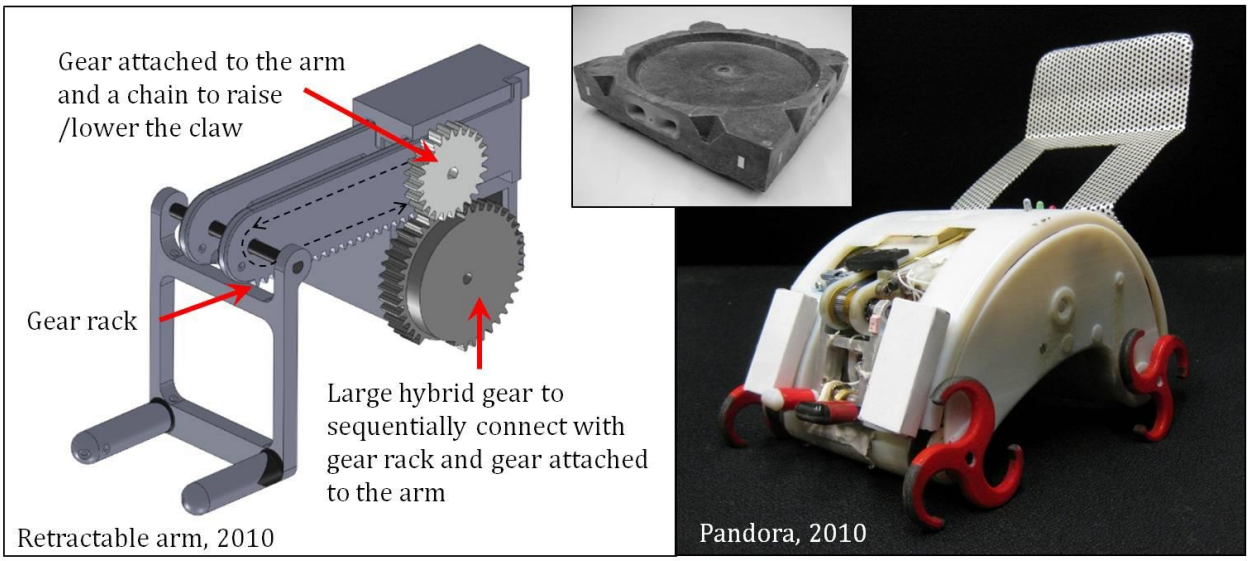


Figure 4.4.c. Left: First version of the retractable arm along with a 3D printed version of the elastic claw. Right: The robot Pandora with a retractable arm. Upper middle: Second version of the brick in cast rigid urethane foam with 3D printed handle.

4.4.2 Final Design

The final manipulator is able to grasp a handle in a brick that is placed on a docking station, rotate it onto the back of the robot for transport, and place it either on top of or next to another brick. The manipulator consists of a fixed arm and a three-pronged claw. The arm module mounts in the front of the robot, and has a geared motor coupled to the claw by timing belt and pulleys, see Figure 4.4.d.a. The arm, claw, and robot chassis have several features to ensure successful docking, undocking and transport of a brick:

- Two prongs are loosely mounted on either side of the center prong and forced inwards by torsion springs. The handle has internal divots for these prongs to snap into to lock the brick securely in place under transport. When the claw is brought down, a fixed wall on the arm

mechanically press open the outer prongs, see Figure 4.4.d.c. This is a simple example of embodied intelligence; When the claw is raised the robot automatically holds on to the brick, when lowered it automatically releases it to ease docking and undocking.

- A fixed prong protruding from the center of the claw helps to stabilize bricks under transport.
- Tactile sensing in the form of a pushbutton on the claw enables the robot to sense if a brick is in possession or not.
- Two micro-switches on the arm enable the robot to sense if the claw is raised or lowered. The claw is normally held in the raised position, and the arm is mounted such that it does not interfere with the procedure of climbing up a step.
- A padded 'shelf' on the back of the robot provides a stable resting position for the bricks while climbing.

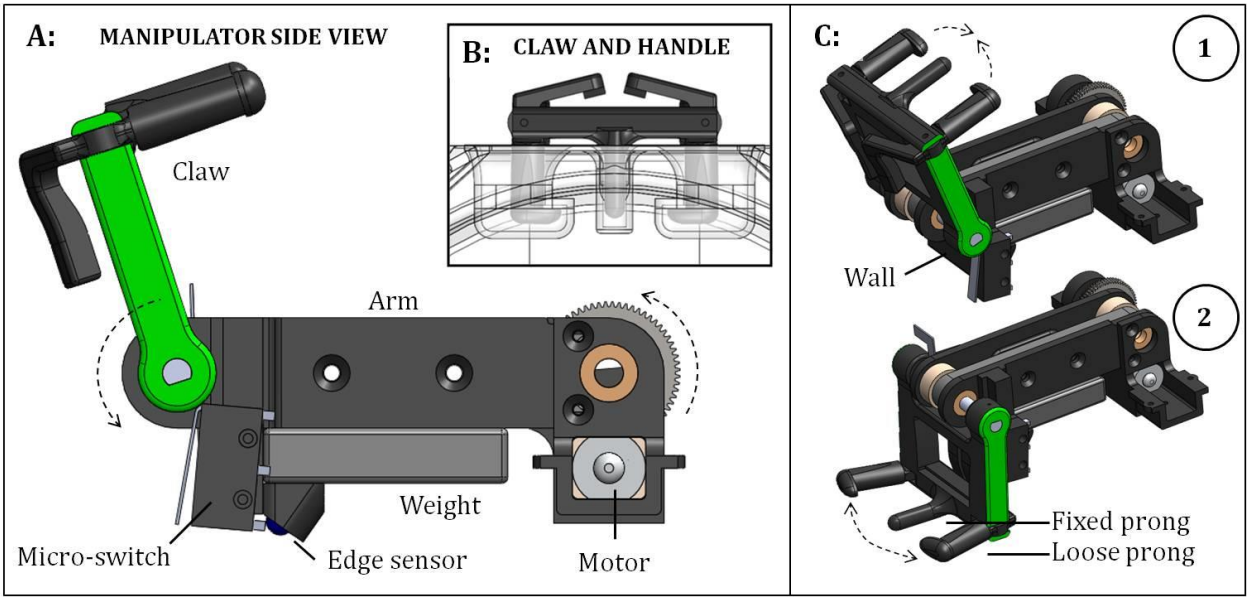


Figure 4.4.d. A: Side view of the arm and claw. B: Top view of the claw latched to a handle in a brick. C: Animation of how the outer prongs automatically disengage from the handle when the claw is lowered against a wall in the arm.

Furthermore, the bricks have features to aid robot manipulation:

- A filleted brick handle to allow docking even if the robot is misaligned (Figure 4.4.d.b).
- Magnets mounted on all six sides of the brick to ease attachment, and mechanical features, inverted from those in the top, were added underneath to make bricks attach like LEGOs.
- As mentioned in section 4.3.1, the width of the cross pattern on the bricks is such that the robot can stop when the front pattern sensors detect white and lower the claw unhindered. The width of the cross pattern in turn determines the position of the pattern sensors.

In section 4.2 the design of the robot body and propulsion system was optimized for climbing tall bricks; now that same robot must be able to climb and descend with and without a brick (Figure 4.4.f). To limit the influence of the weight of the brick on the center of mass, and because batteries and motors for space constraints was placed in the rear of the robot, lead weights were added to five closed compartments in the front of the chassis.

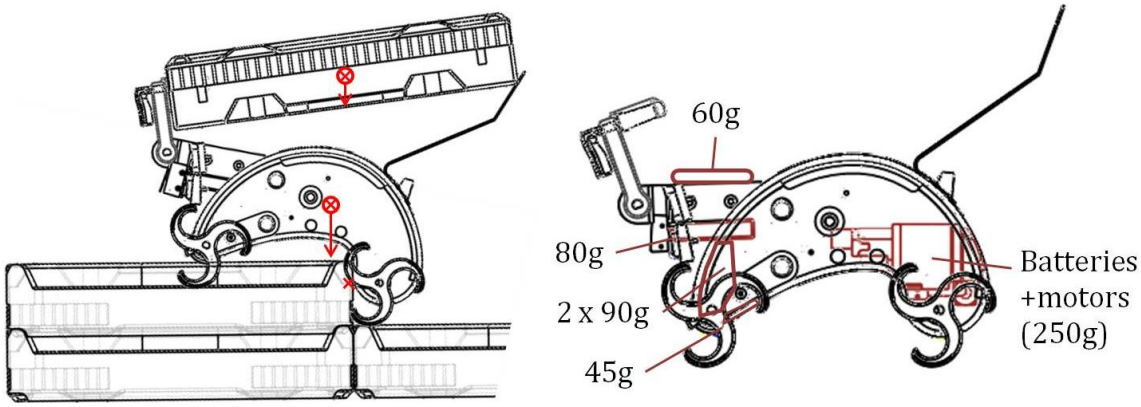


Figure 4.4.f. Left: Robot climbing with a center of mass designed to overcome the clockwise generated torque from wheg rotation and its point of contact with the brick. Right: Lead weights placed in closed compartments in the robot to obtain good climbing properties with and without a brick.

4.5 Coordination

One of the great advantages to the TERMES algorithm is the fact that robots do not have to communicate, but merely avoid each other. Ultrasound transducers already on the robot, described in section 4.3, were reused for this purpose, along with an extra placed in the back; all five upper transducers were used to send out a low amplitude warning signal once a second. Nearby robots picking up such a warning signal with any of their front transducers will stop until they no longer hear this signal. When robots perform ranging, they emit and receive with their lower and upper transducer respectively, causing minimum disturbance to robots on the structure (Figure 4.5.a). When robots emit warning signals they do so with their upper transducers, ensuring that robots on other levels of the structure are also able to hear them.

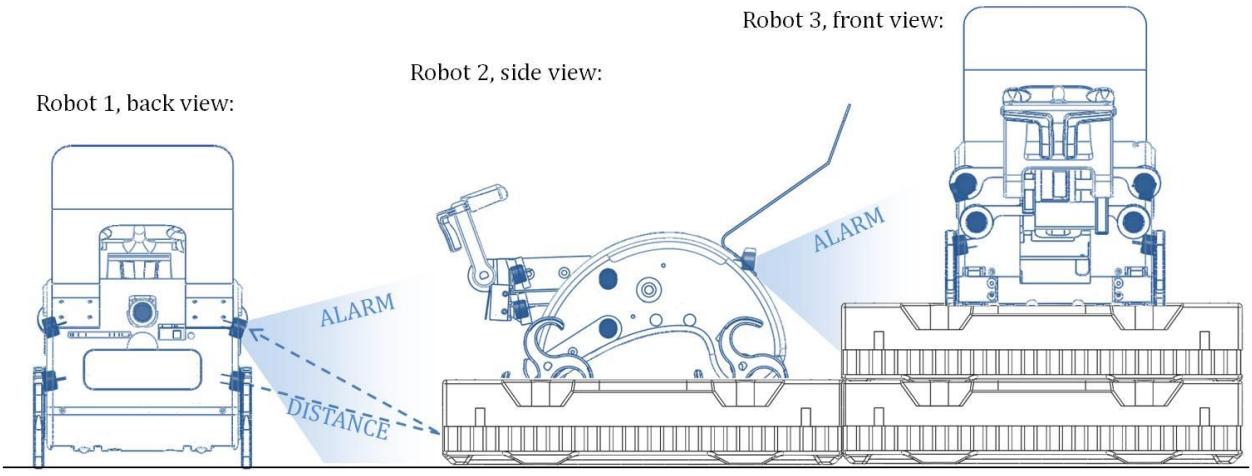


Figure 4.5.a. Position of the ultrasound transducers. When the robot measures distance, the lower and upper transducers are used to emit and receive signals, respectively. The upper transducers emit warning signals; the four in the front are used to detect them.

The strength of the alarm signal and the sensitivity of the robots are regulated depending on their state. For instance a robot about to place a brick needs a bigger safety perimeter than one circling the structure, and a robot about to enter the structure must be extra attentive to robots already on top of the seed brick.

Processing the range signals and polling for alarm signals are time expensive operations and were therefore implemented on a separate processor. This processor performs ranging upon command, and otherwise continuously sends out and listens for warning signals. If a warning signal is perceived a register is set, and remains so for 5s. The main processor of the robot checks this register approximately once a second while circling the structure, and just before proceeding from one brick to the next during operation on the structure.

4.6 Control

Despite the complicated challenge of designing robots able to climb on and manipulate large pieces of material, the use of embodied intelligence makes the control of the TERMES robots relatively simple. Embedded software employs simple sensors to make robot performance reliable over a long sequence of construction. An overview of sensors and general electronics is given in section 4.6.1. Rather than try to make the robot perform perfectly, the control focuses on detecting small scale errors and recovering before they become fatal to the progress of the collective. For instance, robots often slip while climbing up steps, but continuously check their inclination and keep trying until they get it right. Robots also check and correct their alignment with the structure both before, half way, and after passing between two bricks. This process suffers from slow execution, but further enhances reliability. The software architecture is explained in section 4.6.2; it consists of a modular hierarchy which allows easy replacement of sub-routines to work with the iterative process of designing the mechanical properties of the robots.

4.6.1 Electronics

The electronics design is fairly straightforward and does not imply new technology, but is included here for completeness and for future researchers that may wish to use similar strategies.

The main control board of the robot is centered on an ATmega1281 processor running at 16MHz, equipped with a 10-bit ADC, 4 PWM outputs, 6 timers, 64KB of flash, 8KB of internal SRAM, I^2C, and serial ports. This board communicates with an external laptop using Bluetooth, and through I^2C with a separate and simpler ATmega168 processor that handles all ultrasound related sensing and control. Figure 4.6.a gives an overview of sensors and actuators connected to the main processor.

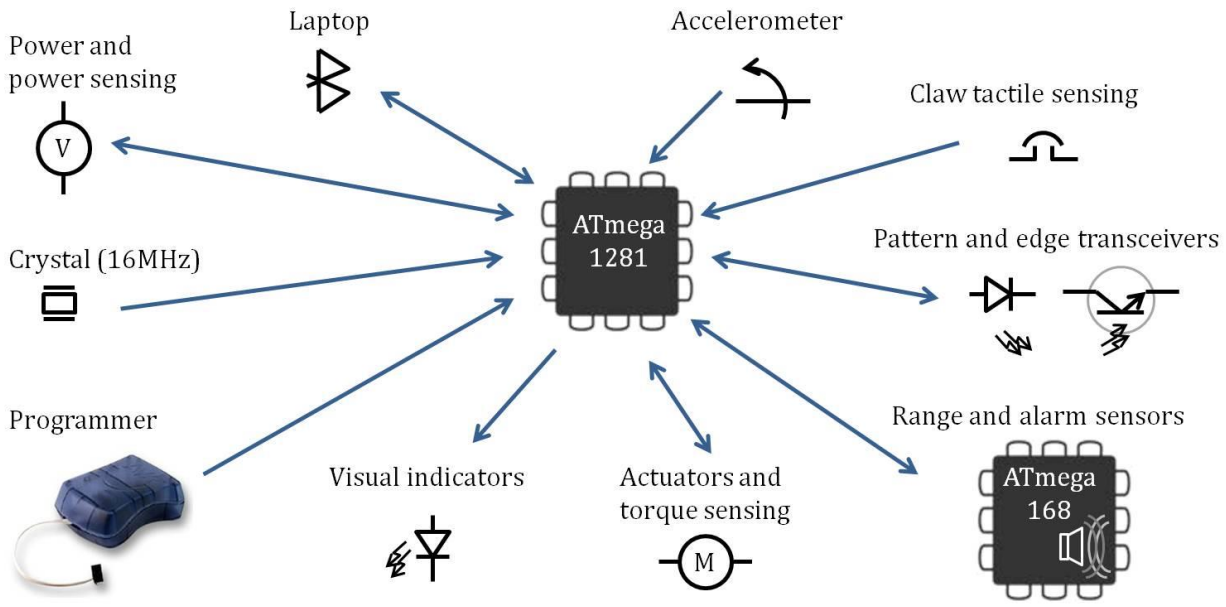


Figure 4.6.a. Block diagram of inputs and outputs from the main controller.

The robot is powered by two 2-celled Li-Ion batteries (750mAh). One battery is used to supply the processor and sensitive signal treatment; another is used only for the actuators to prevent noise coupling (Figure 4.6.b). Depending on the task, the battery which supplies the power for the actuators tends to discharge first, after about 45-60min. The robot is able to detect this and will exit the construction arena autonomously to have its battery pack replaced.

The actuators used are high power micro metal gear motors from Pololu. These are cheap ($16), high power (5kg-cm at 6V, 70mA$_{\text{NO LOAD}}$) motors in a small package (0.94" x 0.39" x 0.47"). All three actuators are controlled by motor drivers, A3953, from Allegro Microsystems, which have a high PWM frequency (55kHz) that can be filtered out to prevent noise coupling to sensitive circuitry and current control to ensure that the motors do not overheat. It is possible for the main processor to measure the current drawn by the actuators, to e.g. enable it to adjust the duty cycle of the driving signal if one wheg is experiencing more friction than another.

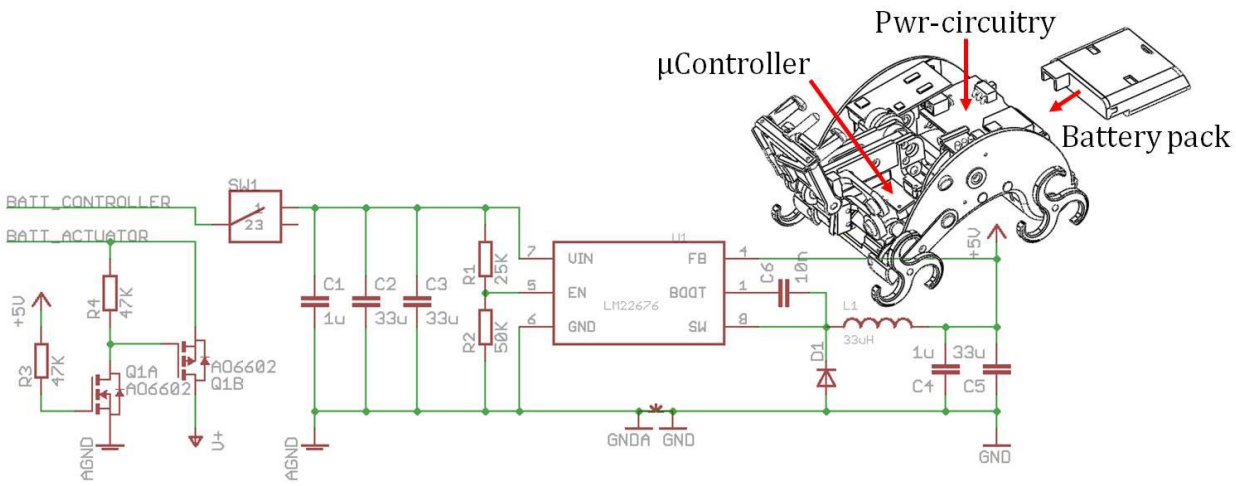


Figure 4.6.b. Robot power circuitry. A switch connects one battery to a 500kHz buck converter which delivers 5V to the controller and noise sensitive electronics; the 5V automatically connects the second battery to the actuator supply (V+). Upper right illustration shows where the micro controller, power circuitry including the motor drivers, and battery pack are located in the robot.

The pattern and edge sensors are based on IR transceivers; an OP298A photo diode and an OP598A phototransistor from Optek Technology mounted next to each other. The driver circuitry for the transceivers is located on the side of the battery shelf (Figure 4.6.c). The emitter is driven by a 400Hz 5% duty cycle signal originating from the main processor; the input from the phototransistor is passed through a two pole Sallen-Key band pass filter with a high Q-value to prevent disturbance from ambient light and high frequency noise. To save power and I/O ports a digital multiplexer allows the processor to choose which transceivers are active, and an analog multiplexer directs the output of the corresponding filter to the processor.

All circuitry related to ultrasound distance and alarm measurements are confined to a separate board centered on an ATmega168 processor running at 1MHz. The circuitry accurately measures objects down to a distance of 12cm by use of speakers (transmitters and receivers, 400ST10P and 400SR10P from Mouser Electronics Inc.) with short ring times, as well as the soft foam ring

described in section 4.3.1 to shield them from direct coupling with each other. To measure distance the controller pings the transmitter; waits 0.6ms to make sure directly coupled noise from the transmitter has died down; then samples the ADC input every 22us up till 2.5ms, corresponding to 10-42cm distance approximately (Figure 4.6.d.b-c). Because of the wobbly gait of the whegs however, the system cannot measure objects accurately at a distance above ~30cm. To accommodate iterative mechanical design, an automatic calibration routine was incorporated into the robot. In this routine the robot brings the claw to the upper position and drives straight on level floor without obstacles for 5s continuously pinging and measuring the response. The maximum response is saved in the EEPROM to be used as a reference for future measurements. To find the distance to an object the reference signal is subtracted from the measured signal, and the beginning (t_{start}) of the first peak (at time t_{peak}) is detected and converted to a distance. The threshold for when a response is considered a peak is altered for near- and far range measurements, as well as if the robot is carrying a brick or not. The latter is due to the fact that the brick footprint is bigger than the robot and tends to cause more direct reflection.

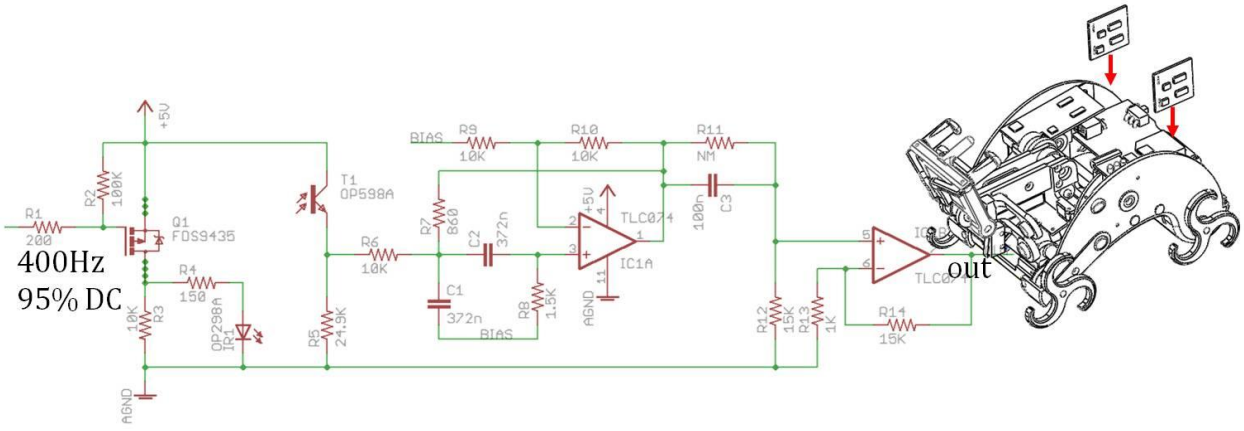


Figure 4.6.c. Driver circuitry for an IR transceiver pair. To limit the need for a negative supply all signals are biased around 2.5V. The drivers for all the IR transceivers (6 pattern sensors and 1 edge sensor) is divided onto two boards that are located on either side of the robot battery shelf.

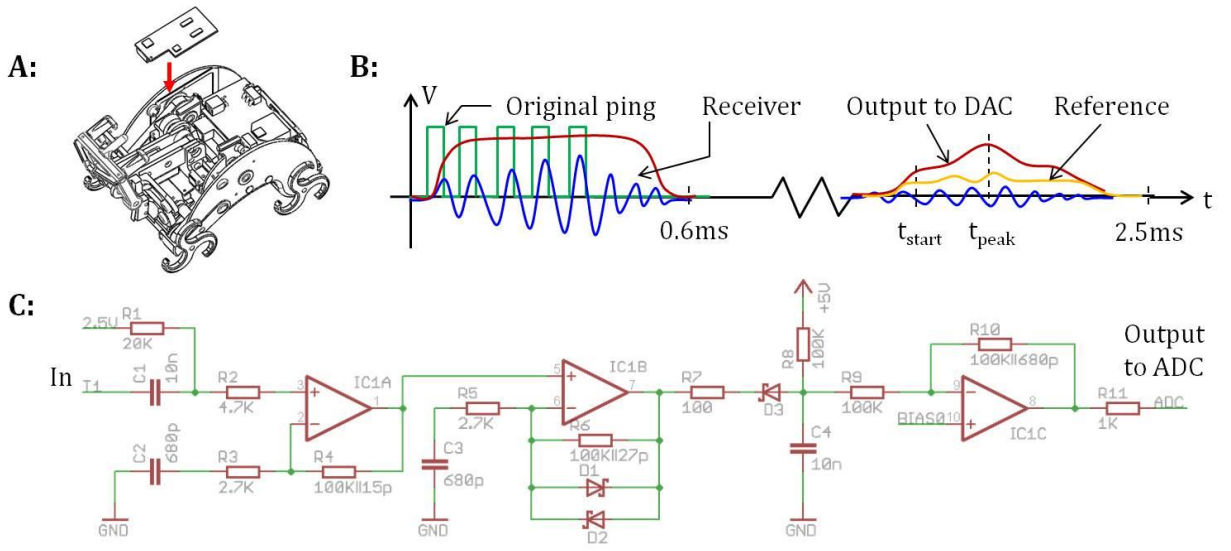


Figure 4.6.d. A: Illustration showing the position of the ultrasound board in the robot. B: Sketch of signals used to measure the distance to an object. C: Input circuitry for a transceiver. The configuration of diodes D1 and D2 amplifies small signals, but not large signals above the diode cutoff voltage. This helps give reflections of small amplitude almost as much influence as large reflections; the amplitude of the reflection does not matter, only the time it takes for the reflected signal to return to the robot. Finally the signal is truncated and low pass filtered.

To measure inclination the robot uses a high-precision 1-axis accelerometer, ADXL103, from Analog Devices. The arm sensors are composed of two micro switches to detect when the claw is raised or lowered, and a manually made switch out of thin brass plate to detect if the claw is connected to a brick or not. Three LEDs (green, red and blue) provides user feedback for debugging. An XBEE module from Digi provides Bluetooth communication with a laptop. The total cost of the electronics, not including the price of the PCBs, is under $400.

4.6.2 Embedded Software

The abstract algorithm provides guidelines on how to make the robots navigate on the structure and decide where to add material in a safe manner; however, the real implementation of this algorithm requires many more details. In order to ease simple modification of sub-routines I implemented a modular software-architecture with three hierarchical layers (Figure 4.6.e). This structure has been fundamental to the iterative process of designing and testing the robot hardware, and translating code from laptop to robots.

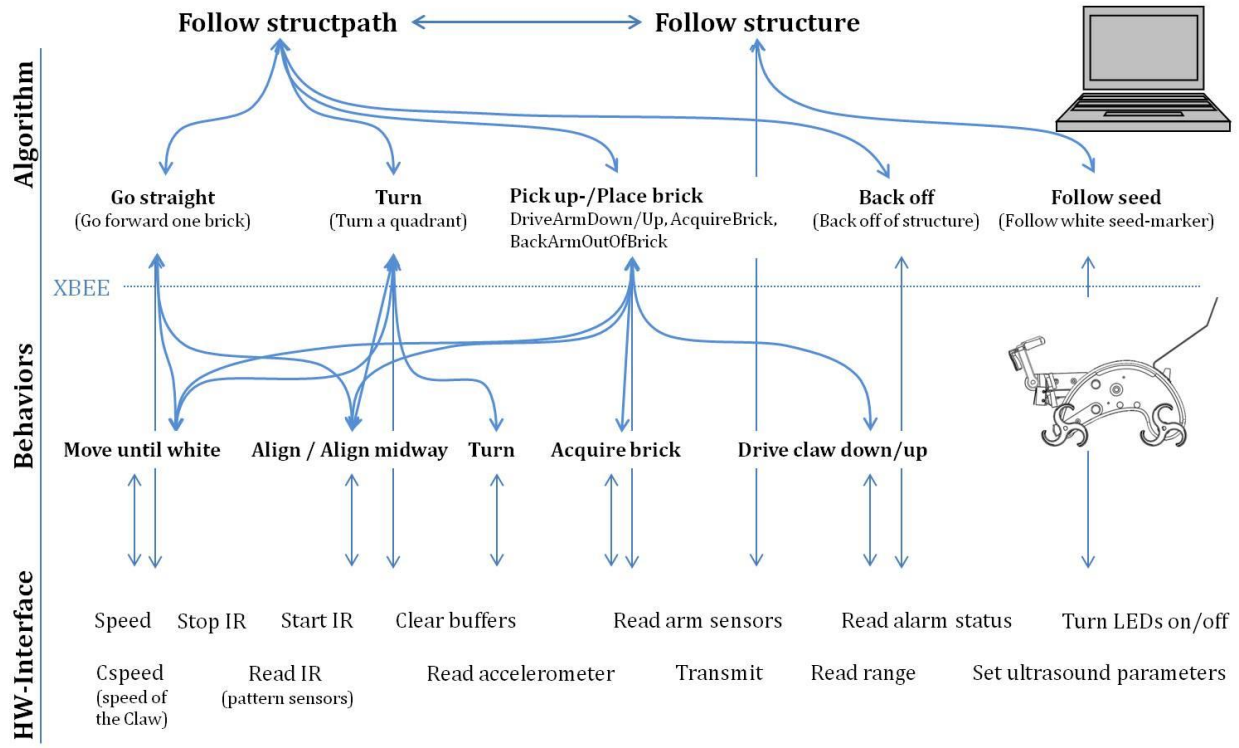


Figure 4.6.e. Software architecture consisting of three layers, divided up between the robot processor and a separate laptop. The algorithmic layer runs sequentially, the hardware-interface layer runs on an interrupt every 50ms, the behavioral layer runs on requests from the algorithmic layer or on replies from requests made to the hardware-interface layer.

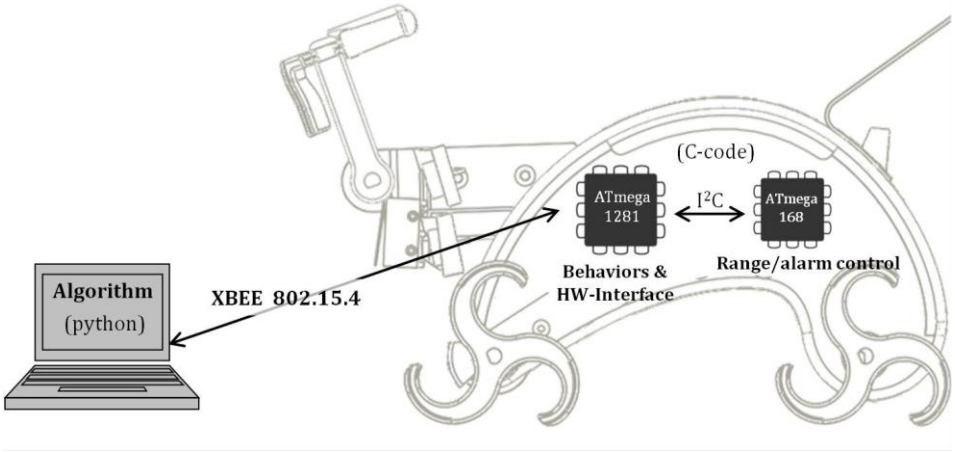


Figure 4.6.f. Processors involved in the control of a TERMES robot. The algorithmic layer of the software is implemented on a laptop, the main processor runs the behavioral- and hardware layers. A separate processor handles all code related to both ranging and alarm signals.

First, to ease debugging, all but the lowest layer was implemented in Python on a separate laptop and commands were sent to the robot via Bluetooth (Figure 4.6.f). With time however, more and more of the behavioral layer was translated to the robot processor. Although there is space left on the processor, to ease debugging, the algorithmic layer, some shell-routines, and the latest additions to the software ("Back off" and "Follow seed") remain on the laptop.

The upper layer consists of the high-level algorithm; it sequentially performs appropriate tasks such as: "Go straight", "Turn", "Pick up", "Follow structure", etc. Each task is dependent on the position in the structpath as well as the outcome of the last task. For instance, if a robot detects a cliff while trying to go straight on the structure, the algorithmic layer adapts by making it back off of the structure (and possibly place a brick) instead.

The middle layer consists of behavioral code, i.e. the code that makes the robot reliably perform each task as ordered by the algorithm. For instance, "Turn" will start the motors and then continuously monitor the pattern sensors until every corner sensor has passed white and sees

black again; afterwards the robot realigns. If at any point the robot gets stuck, detected as seeing the same pattern for a long time, it will perform maneuvers to break free.

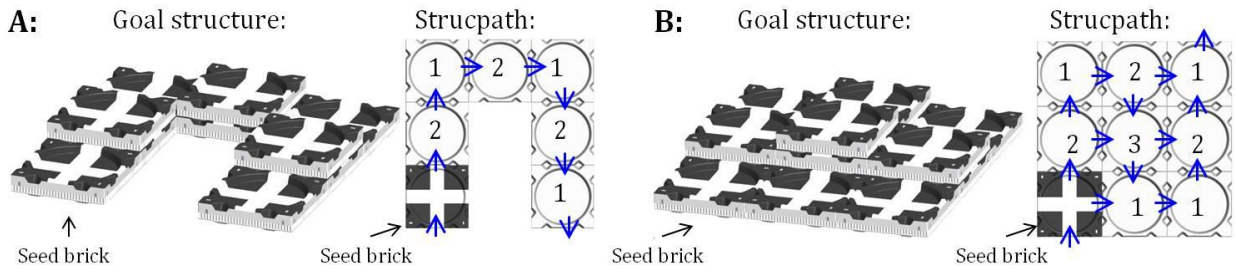
The lowest layer consists of code that directly interfaces with the robot hardware, such as reading and interpreting sensor values, setting actuator values, preparing interrupts, etc.

As mentioned, the algorithmic layer runs sequentially. The behavioral layer runs on inputs from the algorithmic layer or replies from the hardware layer to sensor inquiries. The lowest layer runs on a timer-interrupt; i.e., if activated, sensors are polled every 50ms (an appropriate update rate for the robot the response time of which is around 0.5s). Both the behavioral- and hardware layers have their own circular input and output buffers. Every time a layer is executed it starts by processing the input buffer and then sets/clears flags appropriately. Only sub-routines corresponding to the flags that are set will be executed; this ensures strict control of processor resources. Several hardware-interface routines are, when activated, timed by interrupts.

Behind each of the headings in Figure 4.6.e is code that ensures reliable task execution. The following sections give examples of how the structpath was implemented, and routines which are implemented primarily in either the algorithmic layer, the behavioral layer, or the hardware layer.

4.6.2.1 Implementation of the Structpath

The structpath (section 3.2.1) was implemented as shown in Figure 4.6.g. A simple version of the structpath for demos with single-path structures is a one-dimensional array; each cell composed of sequential XY-positions followed by a number specifying the desired height of bricks in the stack. The first brick is at (0, 0); typically the brick cache is located at (-1, 0); the last cell of the array specifies in what direction robots can exit the structure. Multi-path structures are implemented with two 2D arrays, one specifying the height of each brick in the structure, the other the probability distribution of robot headings at every site in the structure.



C:

$Structpath_A$ = [(0,0,1) (0,1,2) (0,2,1) (1,2,2) (2,2,1) (2,1,2) (2,0,1) (2,-1,0)]

$$Structure_B = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad Structpath_B = \begin{bmatrix} (0.5,0,0.5,0) & (0,0,1,0) & (1,0,0,0) \\ (0.5,0,0.5,0) & (0,0.5,0.5,0) & (1,0,0,0) \\ (0,0,1,0) & (0,0.5,0.5,0) & (1,0,0,0) \end{bmatrix} \leftarrow (P_N, P_S, P_E, P_W)$$

Figure 4.6.g. A-B: Example of single- and multi-path structures from Figure 3.2.c. C: Examples of structpath implementations. Structure A is represented by a simple array in which the first two numbers in each cell are the coordinates of the next site in the structpath and the third number is the desired height of bricks on that site. Structure B is represented by a matrix; each cell contains the probability of a robot moving north, south, east or west. Before moving in that direction, the robot must still make sure that the transition is possible, i.e. that there are no cliffs.

4.6.2.2 "Go straight": Move Forward One Brick

"Go straight" is a piece of software that enables robots to move straight between two bricks and detect if it is a level passing, an ascend or a descend. Currently most code is implemented in the algorithmic layer on the laptop; future work could easily translate this to the behavioral layer on the robot. "Go straight" uses routines in the behavioral layer such as align, align midway, and reverse until white, and uses direct calls to the hardware layer including commands to set the speed of the actuators. "Go straight" is executed every time the laptop receives a response or sensor update from the robot, i.e. once per 50ms.

Figure 4.6.h shows a block diagram of the code. With every sensor update, the accelerometer is checked. When ascending, actuator speeds and requirements for successful alignments are different than if the robot is descending or climbing over level ground. Red arrows in the figure indicate error recovery modes where the robot detects that it has started to, or is continuing, to climb. The blue section is an error recovery mode where a cliff is detected; if properly aligned the reading is trustworthy and the robot returns a 'cliff'-message; if not, it reverses, aligns and tries again.

If the robot has been climbing for a long time it could be caused by improper alignment; if so, it returns to an earlier state, reverses, aligns, and tries again. If the robot succeeds, it returns 'done' along with information of whether it descended, ascended, or passed over two level bricks.

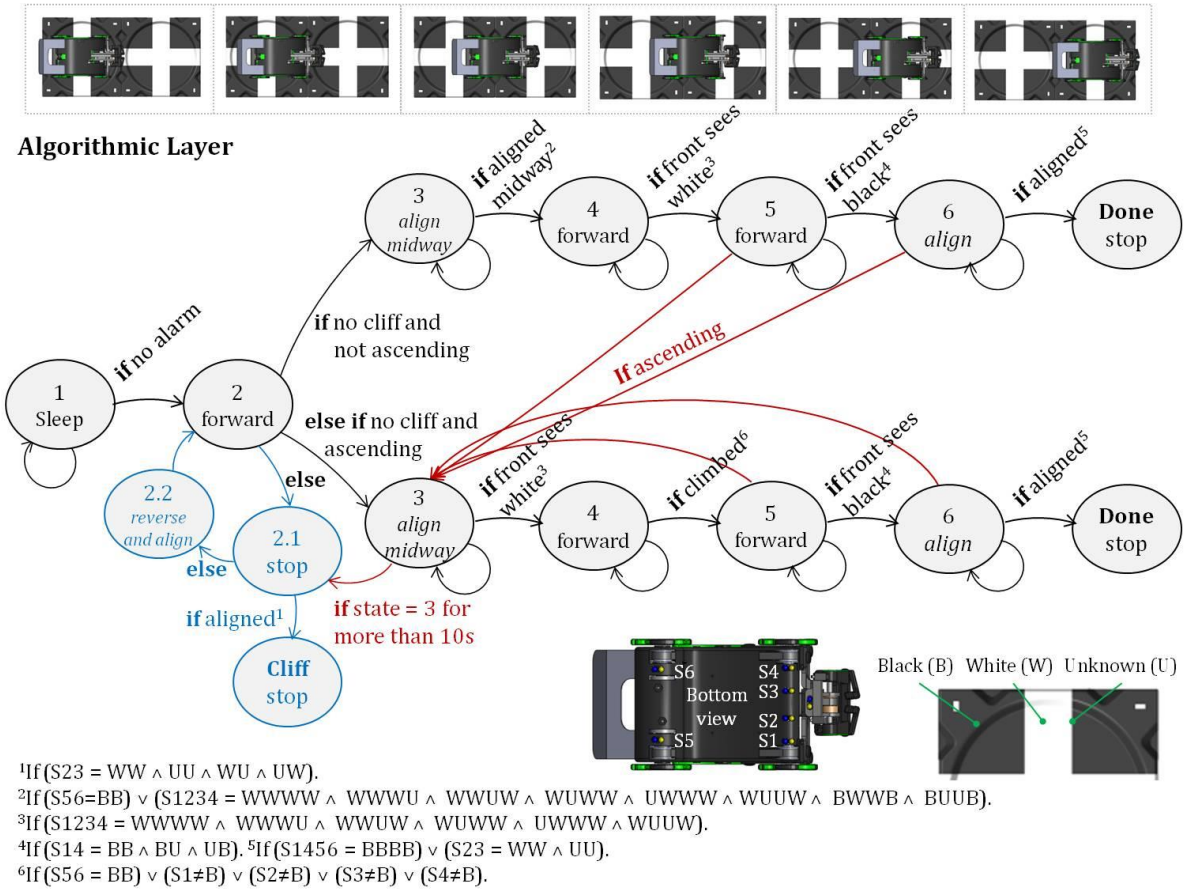


Figure 4.6.h. Block diagram for the routine to move forward one brick: "Go straight". If-statements in the block diagram are clarified by footnotes and illustrations below.

4.6.2.3 "Acquire brick": Navigating the Claw into the Brick Handle

Figure 4.6.i shows the process involved in picking up a brick. The laptop sequentially requests that the robot aligns, reverses until the front pattern sensors see white, brings down the claw, acquires the brick, brings the claw up, and realigns. To make the robot acquire a brick the laptop sends a request to the robot causing the behavioral layer to execute. The behavioral layer sends a command to the hardware layer to start reading in from the pattern sensors. The behavioral layer now runs every time it receives a sensor update, i.e. every time the hardware layer executes with 50ms intervals. It checks if the brick is in possession (if so it stops reading from the sensors, clears the flag which enables the routine, and transmits 'done'), otherwise it keeps trying to acquire the brick. As an extra measure of error correction, if the brick has not been acquired within 6s, the robot reverse, realigns on the brick underneath, and tries the sequence again.

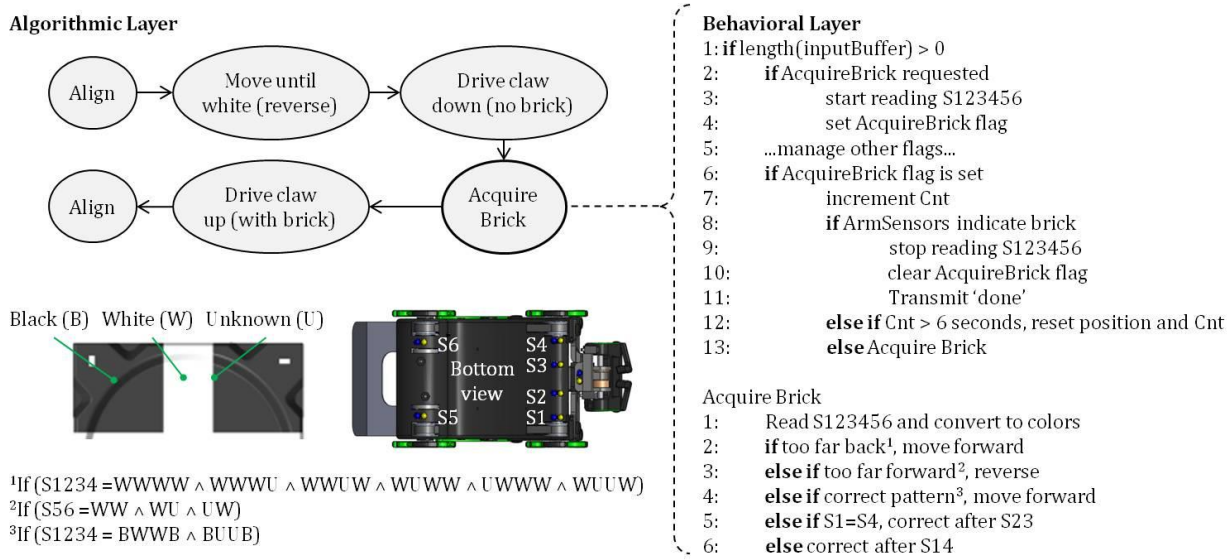


Figure 4.6.i. The block diagram to the left shows the routines called from the algorithmic layer on the laptop to pick up a brick from the docking station. The pseudo-code to the right shows the code related to "Acquire Brick" in the behavioral layer on the robot processor. The illustrations and text in the lower left expands on abbreviations made in the pseudo-code.

4.6.2.4 "Read IR": Read Pattern Sensors

Several routines in the algorithmic and behavioral layers require a continuous stream of inputs from the pattern sensors; they can make requests to do so in the hardware-interface layer or simply read the pattern sensors a single time. Figure 4.6.j shows how this code was implemented; mostly in interrupts for efficient and punctual control.

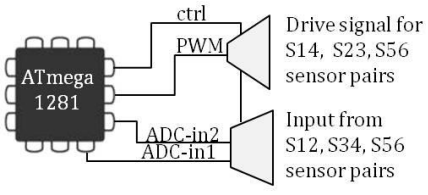
Reading from the sensors is complicated by the fact that the six sensor outputs and drive signals are multiplexed into three channels to save pins on the processer; consequently the first step is to setup these correctly. Next, a timer is started to generate the PWM drive signal. The interrupt from this timer enables a second timer, "IR timer", which triggers a fixed amount of time later and starts the ADC conversion. This ensures that the maximum sensor value is read. When the ADC is done converting, it triggers an ADC interrupt in which the appropriate sensor register and all settings to read the next sensor are updated. When all six sensors have been read a transmit message is placed in the output buffer to be sent to the computer.

```
Hardware Layer
1: if length(inputBuffer) > 0
2:       if readIR requested
3:             set readIR flag
4:       ...manage other flags...
5: if readIR flag is set
6:       Cnt = 1
7:       start PWM timer
8:       set MUX to drive and read from S1 and S4
9:       set ADC to read ADC-in1

Interrupt(PWM timer)
1:       start IRtimer

Interrupt(IR timer compare match)
1:       start ADC conversion
2:       stop IR timer
```

```
Interrupt(ADC conversion finished)
1:       read ADC and update sensor register
2:       increment Cnt
3:       if Cnt > 6
4:             stopPWMtimer
5:             start transmit
6:       else
7:             set ADC to read from appropriate channel
8:             set MUX to drive and read from appropriate sensors
```



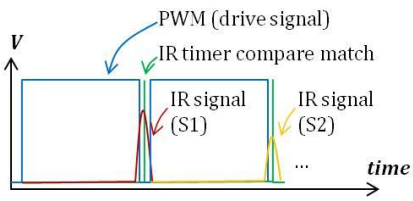


Figure 4.6.j. Pseudo-code for the routine which reads from the pattern sensors. The illustrations in the lower right corner, shows how multiple sensor signals are multiplexed to occupy less pins on the processor and how the drive signal, sensor output, and ADC trigger signal (IR timer) correlates.

4.7 Fabrication

As argued in 4.1 easy fabrication is vital to a successful implementation of the system, because it eases the iterative process of designing system mechanics. I can produce a new robot from scratch in a week; producing a new brick takes about 30min. With automatic assembly of PCBs this process could be sped up considerably. The size of the robot is 175 x 110 x110mm (excl. the brick shelf) and it weighs about 800g; the bricks measure 215 x 215 x 45mm and each weigh between 210-240g. A total of 9 robots and about 90 bricks were produced; 3 and 50 of the final versions respectively.

The robot mechanics is composed largely of standard off-the-shelf electronics and 3D printed parts: the chassis consists of 3 parts and 4 whegs, the arm of 2 and the claw of 3 (Figure 4.7.a.a). The arm, chassis, and claw are separate modules and each can be modified without affecting the others. Assembly is largely done by slotted features; the robot requires a total of 6 screws, and an additional 8 to mount PCB's. Bearings are pressure fit into sockets in the chassis, and all axels are designed to have large tolerances so that they can quickly be manufactured and modified by hand.

The robot electronics consists of 5 PCBs, one for the main processor, one to control the ultrasound, two to control the IR sensors, and one for power electronics. All sensors and actuators are pressure fit into the chassis. For future versions, I would recommend switching to a different wiring scheme between PCBs as the current connectors and wires are fragile and tend to break easily when the chassis is modified (Figure 4.7.a.b). Automatic calibration routines where coded for both pattern and ranging sensors to allow quick updates of thresholds incorporated in software.

The bricks are cast in 8lbs rigid urethane foam using a specialized silicone mold with a wooden backing (Figure 4.7.b). The original bricks had a 3D-printed top shell to prevent wear and tear from the robots, but later versions are covered in Styropor1000 which provides a shock resistant hard coating that sticks to the oily surface of the urethane. All parts not consisting of foam is inserted into the mold before the urethane is poured; magnets snap onto magnets embedded in the side of

the mold, the 3D printed top is placed in the bottom, and the brick handle is screwed onto the side of the mold. The cost of a robot and a brick is estimated around $1500 and $25 respectively.

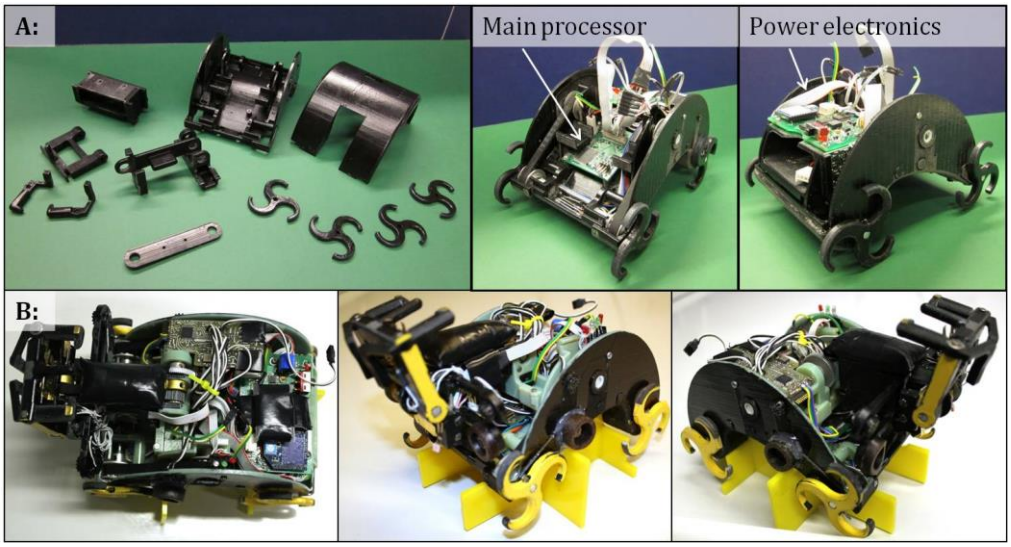


Figure 4.7.a. A: Assembly of Khali 2011. B: Isis 2013, showing wires between PCBs in the robot.

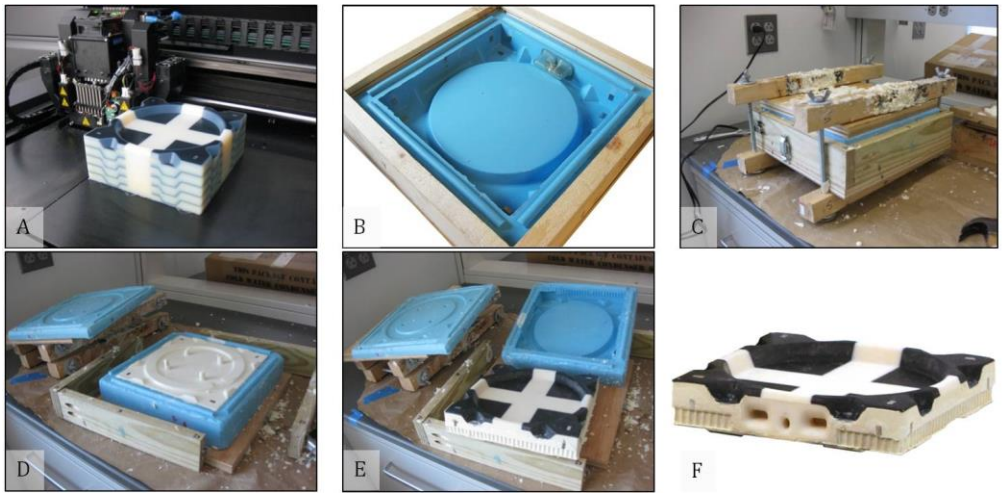


Figure 4.7.b. Snapshots of the production process: a) 3D printed tops later replaced with a layer of scratch resistant paint, b) bottom half of the mold with magnets and handle, c) clamps holding the wooden frame as the inner urethane foam expands and sets, d) after about 20min the foam has set and the top of the mold can be removed, e) brick removed from the mold, f) the final brick.

4.8 Performance

This section reports on performance and failure modes of the implemented TERMES system. At the current stage the algorithmic framework is not tolerant to errors; instead the success of the system is dependent on how reliable every behavior is implemented on the physical robots. The following sections describe experimental results related to each of the sub-challenges locomotion, manipulation, and navigation, as well as performance metrics and failure modes of the final system.

4.8.1 Experimental Results

I conducted many experiments and built many different structures with the TERMES system; the following text describes some highlights with the most recent versions of the robots. Each construction-related experiment represents hundreds of sequentially completed sub-tasks using robots restricted to on-board sensing and represents a large step forward compared to other multi-robot construction systems (section 2.3); nevertheless, more work is obviously needed to perform reliable long-sequenced construction without human interference.

4.8.1.1 Locomotion on Different Surfaces

A propulsion system based on whegs enables the robot to traverse rough terrain (Figure 4.8.a). To test this ability the time taken to travel 1.2m on six different surfaces over five trials was measured: linoleum (9.5±0.5s), carpet (10.2±0.1s), pebbles of size 25±10mm (14±1s), grass (14±1.7s), mulch (11.0±0.4s), and snow (18.0±3s). Although the sensors of this robot will not allow it to navigate outside and the current bricks need to be placed on a fairly regular surface, the robot mechanics are perfectly adequate for the task.

Figure 4.8.a. Exploiting whegs, the robot is able to traverse various surfaces with simple control.

4.8.1.2 Navigation on the Structure

The biggest concern with respect to performance is navigation on the structure, because the consequence for failure is much more severe compared to failures in navigation off the structure. A robot which falls off the structure can be severely damaged, and worse; a robot getting stuck on the structure risks stopping the entire progress of construction. In contrast a robot which fails to find the seed marker will simply continue around the structure, or in the worst case lose sight of the structure entirely still without impeding the progress of the collective.

To test navigation on the structure a robot was set to travel the structure seen in Figure 4.8.b, 20 times with and without a brick. The total number of tasks executed was 240 moves from one brick to the next (including 80 ascents and 80 descents), 244 ninety degree turns, 402 fine alignments on top of bricks and 80 alignments midway between two bricks. The robot completed all tasks without errors, i.e., the robot always correctly kept track of its actual movement and never failed to move between bricks or turn as intended. Completion time was approximately 2hrs and 30min.

Figure 4.8.b. Structure traversed 20 times with/without a brick to check navigation consistency. Time of completion was around 2hrs and 30min; the battery was replaced halfway.

4.8.1.3 Manipulation of Bricks

A performance test was conducted to evaluate the reliability of the manipulation behaviors. In this test a robot moved a brick from point A to point B shown in Figure 4.7.c twenty times in a row. The total number of tasks executed was 81 fine alignments on top of a brick, 40 ninety-degree turns, 20 brick acquirements, 20 brick lifts, 20 brick placements, 20 brick disengagements, and reversing 60 times until the front sensors saw white. The latter is hard because the robot must reverse straight independent on wheg positions and without falling off of the structure, but far enough that the claw can be raised or lowered without hitting the brick in front of the robot. All twenty laps were completed in 17min without errors, taking an average of 15±5s to pick up a brick, and 24±5s to place it.

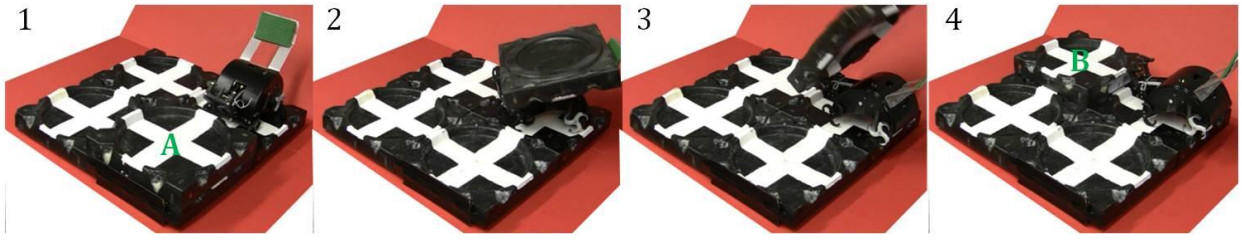


Figure 4.8.c. Consistency test of manipulation; a robot moved a brick from point A to B 20 times in a row with no errors. Time of completion was 17min.

Figure 4.8.d. Kali 2011, constructing a 10 brick staircase autonomously on top of a layer of bricks using a fixed set of commands. Time of completion was 24min.

4.8.1.4 Single Robot Construction on the Structure

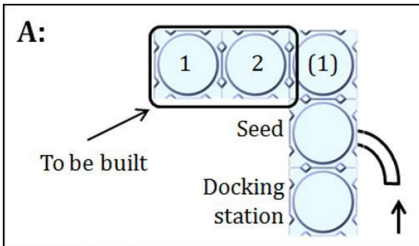
Before the algorithm was implemented on the robot and before the robot was equipped with ultrasound sensors to navigate off the structure, I had it complete a 10 brick staircase on top of a layer of bricks (Figure 4.8.d). This process took a total of 24min, consisted of 106 sub-tasks, all of which were completed successfully due to low-level error recovery. This staircase represents the biggest structure ever built by the TERMES system; more than 18 times the volume of a robot.

4.8.1.5 Single Robot Construction: Full Implementation

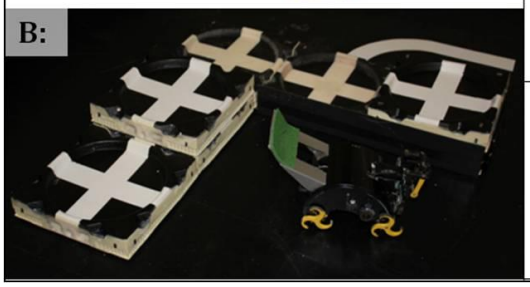
I evaluated the consistency of the fully implemented system by testing the ability of a single robot to complete a three-brick staircase 10 times in a row (Figure 4.8.e). All errors which the robot was not able to correct on its own were recorded. Minor errors, such as the robot getting stuck during turns, were manually corrected during runtime; three such errors occurred. Additionally the robot failed to align properly midway once. Larger errors were corrected by turning the robot off

and returning it to the seed marker without a brick. The larger errors occurred due to 1) failure to detect a low battery, 2) dust accumulated on pattern sensors, 3) internal I²C bus communication failure, 4) alignment error causing to the robot to fall off the structure, 5) poor brick placement on the structure, and 6-7) edge detection errors where the robot mistakenly either identified a brick as the end of an existing structure or mistakenly saw a brick where there was none. To complete this experiment the robot travelled more than 80m. Each 3-brick structure took 20±5min to complete. When moving (i.e. discounting picking up and placing bricks) the average speed of the robot was 0.76m/min on top of the structure and 0.74m/min off it.

A:



B:



C:

Task	No. of Attempts	Successes	Failures
Pick up a brick	36	36	0
Attach a brick			
On ground	20	18	2
On top of structure	10	9	1
While on ground			
Follow perimeter to seed	41	41	0
Detect seed	41	36	5*
While on structure			
Align on top of brick	342	341	1
Align midway	70	69	1
Turn 90°	240	236	4
Detect brick edge	36	34	2

Figure 4.8.e. Consistency test with one robot building a 3-brick staircase 10 times a row. A: Schematic of the structure to be built in extension of the seed brick, the docking station and a fixed brick (1). B: Photo of completed structure. C: Successes (including error recovery) and failures of sub-tasks in the experiment. The total process took about 4hrs; the battery was replaced twice.

*: If the robot misses the seed marker it does not constitute a severe failure, but simply causes the robot to circle the structure an extra time.

5.8.1.6 Multi-Robot Construction

The TERMES robots separately and collectively constructed a multitude of staircases, walls, semi-enclosures, and platforms several times their own size. Figure 4.8.f.a shows three robots completing a partially built castle structure autonomously without errors; requiring a total of 9 brick pick-ups and placements, 278 alignments and ninety-degree turns on top of the structure, 52 ascents and descents, 25 crossings between bricks the same height, and a total travel distance of over 30m in 23min. Figure 4.8.f.b shows three robots building an 8-brick trident structure from scratch; here each robot is given a separate branch to complete. This required 16 brick pick-ups and placements, 145 alignments and ninety-degree turns on top of the structure, 15 crossings between bricks the same height, and a total travel of over 20m in 31min. The added material in the trident is more than 14 times the volume of a robot; again this structure was completed autonomously without errors. These demonstrations were mainly designed to show the complete system working autonomously with multiple robots using only on-board sensing; nevertheless there are still many future challenges to solve before such demonstrations could be done repeatedly and reliably. I discuss these issues in the next section.

4.8.2 Failure Modes

The TERMES system is limited to operation on smooth, level black floors, steady light conditions, and low noise environments. Within these boundaries, as the previous section shows, it can produce structures many times the volume of an individual robot autonomously; the only limit to the size of the structure is the reliability of the system.

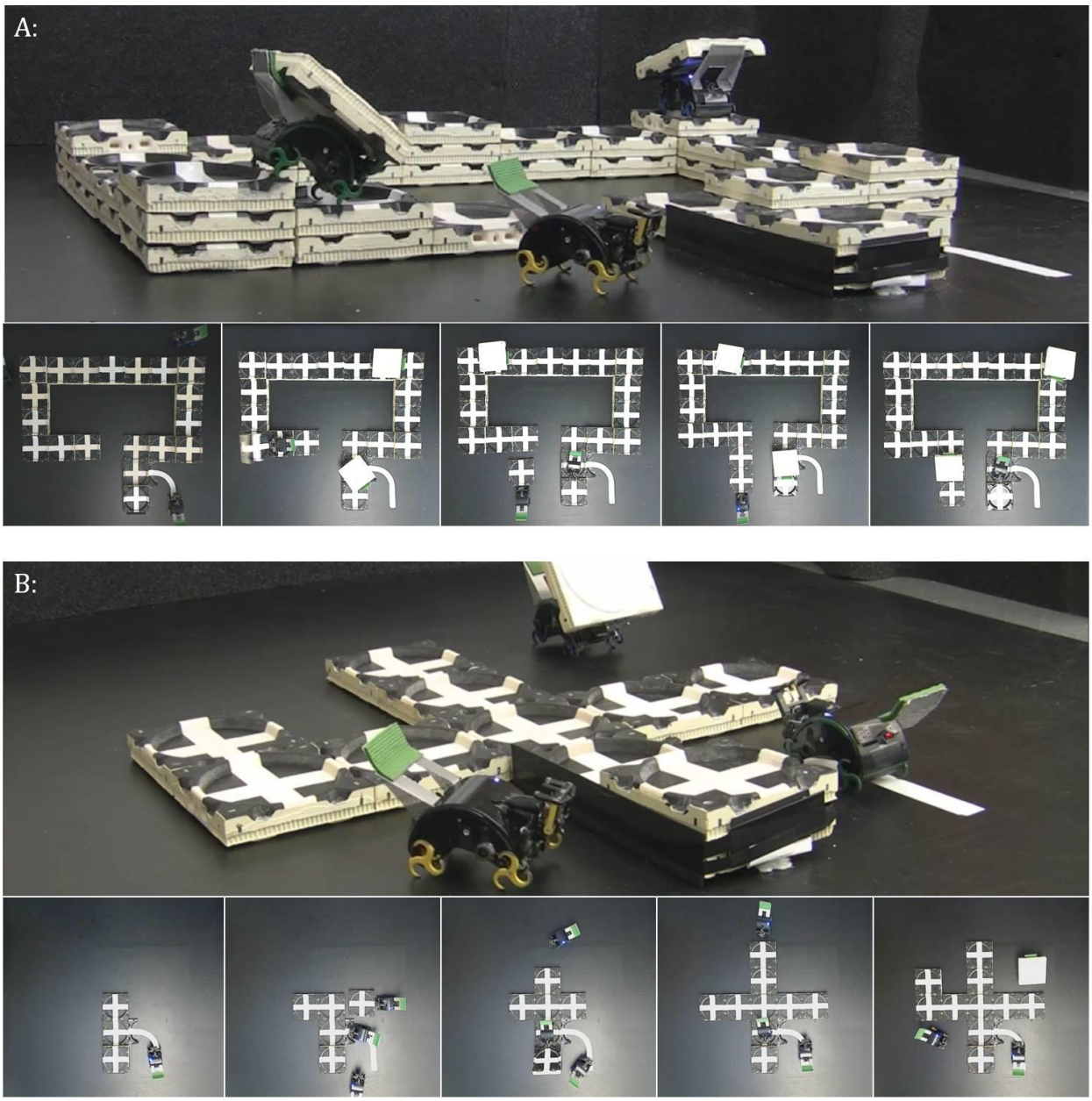


Figure 4.8.f. Autonomous construction with three robots, 2013. A: In this experiment the robots add the last four bricks to complete a castle-like structure. This took a total of 23min. B: Each robot is given a separate branch of a trident to complete, completion time was 31min.

Though rare, the most common 'low-level' errors in the system are improper brick placements (~10%), robots misjudging the end of the structure (<6%), and robots climbing out of the brick bowl indentation while turning and becoming stuck (<2%). The failure rate is low because of the focus on error tolerant control; however, during a long sequence of construction these errors are bound to happen, leading to improper brick placements and failed robots possibly in the direct path of the structure. Some of these errors can be fatal to the entire progress of the system, whereas some will only be an inconvenience. For instance, improper placement of a brick which creates a cliff will hinder all robots from progressing along that path, but for multi-path structures that may only affect part of the structure. Another example is a robot which is poorly aligned on the structure; if it becomes stuck it could create an obstacle for other robots, however, if it falls off the structure the rest will still be able to carry on and simply circumvent it on their way back to the seed brick.

Other risks are 'high-level' navigational errors that cause robots to end up in positions different from where they think they are and therefore place bricks in undesired positions and/or travel the wrong way down the structpath. Such a mistake might be detected if the robot experiences a cliff where there is not supposed to be one; if so the robot can perform a greedy search to get off the structure.

In general, however, to complete larger structures autonomously we need a more error tolerant approach. Future work should focus on implementing error tolerance on the algorithmic level as well, allowing the system to deal with obstacles like a stuck robot or an improperly placed brick.

4.9 System Extensions

Several small-scale improvements could be made to TERMES system including automatic brick dispensers, automatic robot chargers, and making the robots able to sense whether or not they successfully attached a brick (currently brick placement is an open-loop operation). A more involved improvement would be to make the robots able to detach bricks from the structure. This would increase the set of admissible structures and also enable robots to fix some of the failure modes mentioned in section 4.8.2. Using bricks labeled with positional information (such as RFID tags) could prevent navigational errors, enable high-level error recovery, and prevent the need for a single seed brick. Enabling robots to navigate between several docking stations could further prevent bottlenecks by allowing multiple robots to acquire bricks and enter the structure at the same time.

The successful implementation of the TERMES system is mostly due to the highly optimized interplay between robots and bricks. The downside of this design choice is that the robots are strongly dependent on these specific bricks and can operate only in relation to them; the robot-to-brick interface must remain the same. The following sections discuss possible brick modifications that do not affect this interface; such as bricks that mechanically unfold to produce roofs and windows when attached to the structure (section 4.9.1) and smart bricks (section 4.9.2). With simple additional sensors robots could be made more tolerant to changing light conditions and noisy environments, as would be necessary in outdoor environments.

Other improvements would require a full redesign of the hardware and algorithm. Making the robots able to climb up straight walls, for instance, would completely eliminate the need for staircases and would prevent issues with improperly placed bricks and accidental cliffs. Heterogeneous robots might deal with special tasks, such as specialized robots able to remove failed robots on the structure. The use of heterogeneous bricks could further improve the set of

admissible structures. However, arguably the most useful system extension is to make robots able to sense obstacles (dead robots or alien objects) and the algorithm able to deal with such issues, e.g. by building around them. In section 4.9.3 I present an exploratory study of how to make the current system produce a structure adapted to an obstacle of unknown height in the environment.

4.9.1 Expanding Bricks

System efficiency could be improved dramatically with expanding and collapsible bricks. These could be standard sized bricks which collapse to permit a robot to carry more than one at a time, or standard sized bricks which expand to fill the space of several bricks. This section describes some pointers for the design of bricks to be manipulated by the current TERMES robots (i.e. modifications that do not alter the robot-to-brick interface). Extending the algorithmic framework to work with heterogeneous bricks is still an open challenge.

Besides adding more volume with every deposition, expanding bricks could serve several structural purposes including roofs (not load bearing), columns for windows, and even bridges to span gaps in the structure (Figure 4.9.a). Several properties must apply to these bricks:

- The robot-to-brick interface must remain unaltered.
- The weight of the bricks cannot exceed 240g; above this weight, the robot will have difficulty climbing, and may tip forwards when attempting to pick up a brick.
- If load-bearing, the attachment of the bricks to the structure must be appropriately strong.
- The expansion mechanism should be passively activated, e.g. by springs.
- The expansion should be passively triggered as the robot places the brick on the structure.

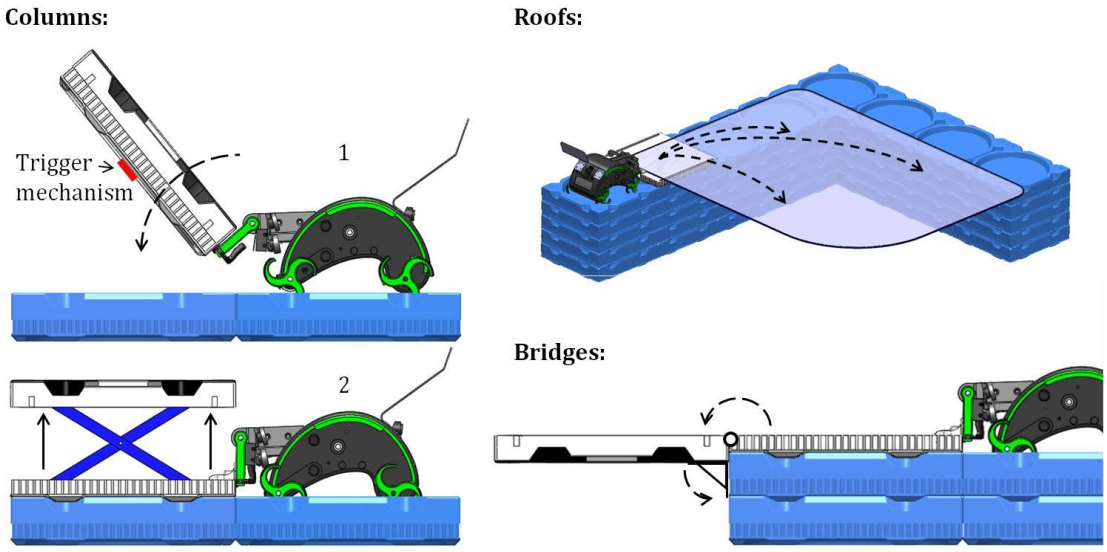


Figure 4.9.a. Bricks which automatically expand when placed to form columns, roofs, or bridges.

4.9.2 Smart Bricks

As mentioned in the beginning of section 4.9, smart bricks can extend the functionality of a structure and may simplify the design of the robots. Simple RFID tags on the bricks can enable robots to label reference positions on the structure, decrease navigational issues, and limit the need for a single seed brick. More capable bricks might be used for many other purposes, e.g. to measure structure loads, temperature, or humidity.

Here, I present brick hardware that enables it to activate and deactivate its magnetic attachment to neighboring bricks. Robots can by touch request that bricks turn off their magnets, thus making it feasible for them to detach bricks from the structure. Smart bricks obviously come with the trade-off of additional cost and complexity.

The magnets in the bricks are replaced by electro-permanent magnets, EPMs [88] (Figure 4.9.b.a-b), composed of a hard- and a semi-hard magnet surrounded by a coil. The EPM is turned on and off simply by temporarily sending current either way through the coil to reverse the polarity of

the semi-hard magnet. The holding force of the EPM designed is 318g; on the same scale as the holding force of the passive magnets currently embedded in the bricks. The circuitry shown in Figure 4.9.b.c was developed to control up to 8 EPMs, as well as 8 LEDs, a speaker, and an optional XBEE link for Bluetooth communication with a separate computer. It is based on an ATmega1281 and powered by a 3.3V battery, converted to a 5V electronics supply, and a 20V supply used to charge an OSCON 180uF capacitor. The charge stored in this capacitor is used to quickly deliver the high current needed to switch the EPMs. The circuitry can be turned on by temporarily connecting two electrodes, e.g. mounted on the surface of the bricks. Once connected, the processor keeps itself powered for as long as desired. Using non-volatile memory, a single touch by the robot can make the brick activate the magnets, and two make the brick deactivate them. With small modifications of the robot, the brick speaker and/or LEDs can be used to provide near-range feedback between the robot and the brick to inform it if the magnets are on or off.

The total weight of the hardware is ~84g, light enough that the current robot should be able to lift a smart brick without modifications. Although the brick worked well, I did not have time to implement and test the necessary routines on the TERMES robots.

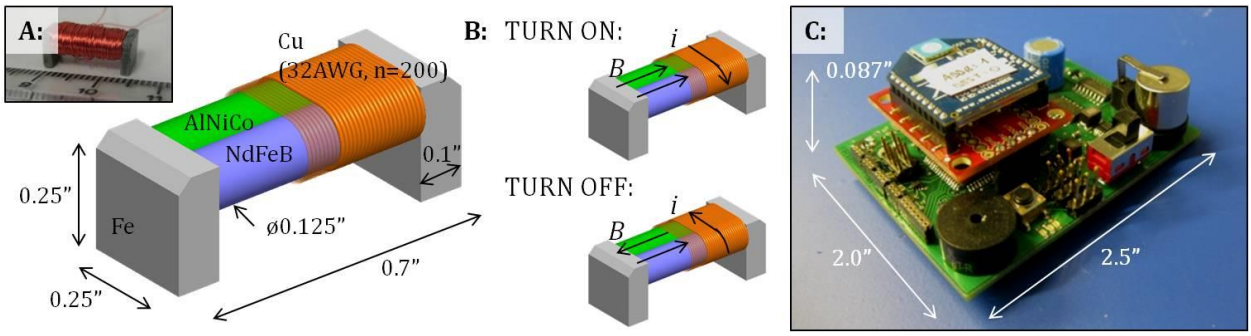


Figure 4.9.b. Smart brick components. A: Design of electro-permanent magnet (EPM) with a holding force of 318g when attached to a 10mm thick iron rod. B: How to turn the EPM on and off. C: Circuitry developed to fit in a smart brick with the ability to control up to 8 EPMs.

4.9.3 Adaptive Structures

As mentioned in the beginning of section 4.8, the ability to produce environmentally dependent structures would greatly improve the robustness of the system. This section shows a simple extension to the system that lets the robot build a structure necessary to bring it to a goal marked by a white surface. An extra IR sensor was added to let the robot detect walls in front (this test was performed before the robot was equipped with ultrasound sensors to navigate off the structure).

The robot starts out knowing only its position with respect to the seed brick and the docking station (Figure 4.9.c). It then picks up a brick and performs a greedy search of where to go, i.e. it turns ninety degrees to the right, discovers a brick and moves on to it. Next, the robot attempts to go straight, but discovers and edge, turns ninety degrees to the right, sees a brick and moves on to it. This procedure continues until the robot discovers a wall. The robot deposits its brick and climbs it to check if it can now scale the wall; if not the robot returns to the docking station for more bricks. The robot continuously updates its map of the structure. Finally, when possible, the robot climbs the wall, and finds itself on the white surface at which point the goal is reached and it stops. The robot completed the task in 23min without errors; it reached its goal by building a structure 11 times its own volume.

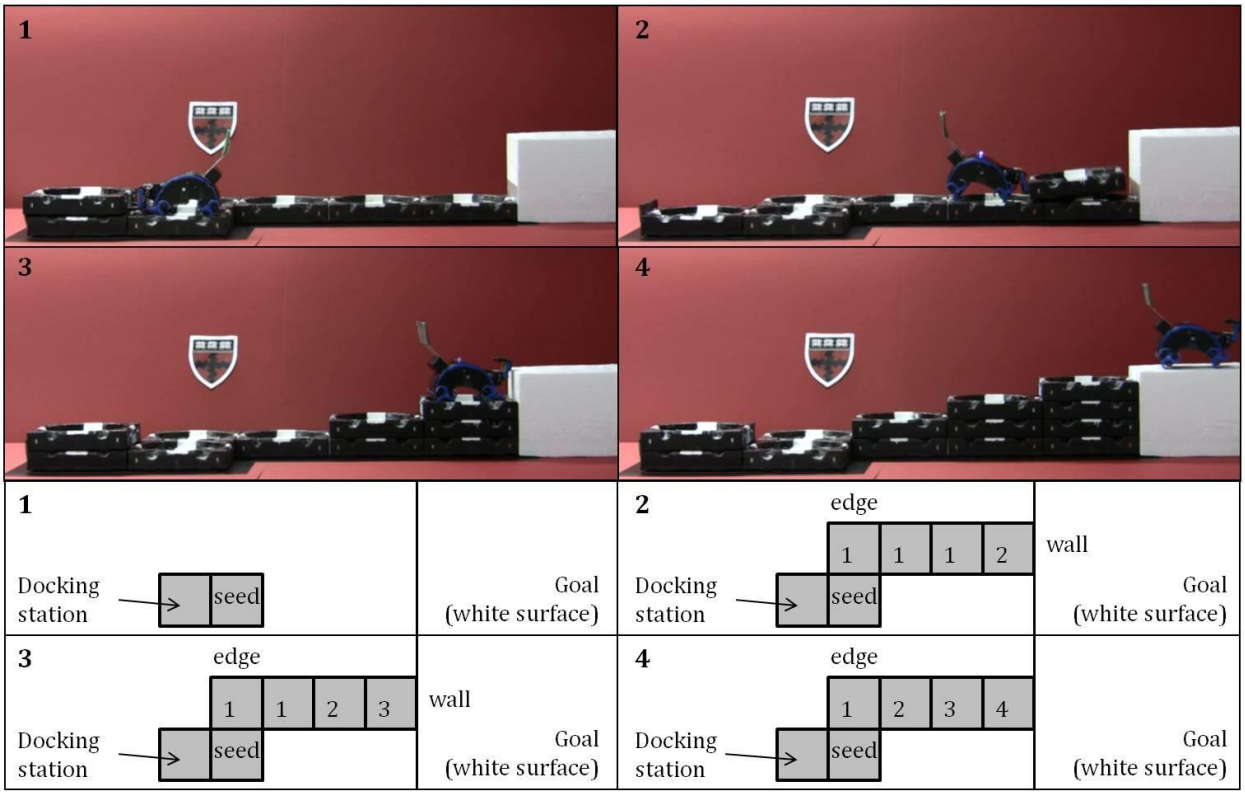


Figure 4.9.c. Khali 2011, constructing a structure that allows it to move to the goal, consisting of a white surface. The top panel shows snapshots of the process; the bottom panel shows how the robot perceives its environment at each of the corresponding snapshots.

Chapter 5. Macrotermes

As described in Chapter 1, termites construct intricate mounds and nest structures to suit the needs of their colony; yet coordination does not rely on central control, but presumably emerges from the combined actions of comparatively simple individuals restricted to local sensing. This approach is incredibly scalable, working for colonies of small numbers to millions of termites, and is remarkably fault tolerant and adaptive to environmental changes; all features we would like to co-opt in the design of robotic construction crews that need to work in a real-world setting with minimal human interference. Unfortunately, relatively little is known about how termites coordinate construction on scales so much larger than themselves, and how they utilize both local and environmental stimuli to create functional structures. In fact, while termites have often been the inspiration for studying social insect construction, much of our current understanding comes from the studies of wasps and bees that construct in small groups and are more easily observed.

Here, I present exploratory work towards the eventual goal of understanding the driving factors behind coordinated construction in termites, especially (1) what stimuli guide the construction process, (2) what local choices do individual workers make that result in a global functioning structure, and (3) if all workers engaged in construction are exhibiting the same behavior?

As a secondary contribution to my thesis, I have developed methods and tools to enable collection of more high-resolution and quantitative data on termite construction than has been feasible in the past. This includes software to perform manual and semi-automated tracking of position and orientation of individual termites confined to experimental arenas using overhead camera recordings, and software to semi-automatically assign behavioral states to each termite. Using these methods and tools I have produced some initial hypotheses on differences in behavior related to cement-pheromone stimuli [27], differences between individuals, and differences between two termite species that, despite morphological similarity, build very differently shaped

mounds in the same environment. These results are preliminary, but will form the basis for future more rigorous experiments. In addition, I co-developed several other experimental tools and methods, including 3D scanners to automatically record detailed soil movement in experimental arenas [26], and mound-insert observation chambers to record building progress during repair in-situ.

Section 5.1 introduces the mound-building termites in more detail and section 5.2 describes related work in studies, methods, and tools to survey termite construction. Software tools to track motion and behavior of individual termites in confined experimental arenas are described in section 5.3. Exploratory studies of cement-pheromone based on these tools are presented in section 5.4. Section 5.5 describes current progress on 3D scanners to record detailed termite construction, and section 5.6 methods to observe repair in-situ.

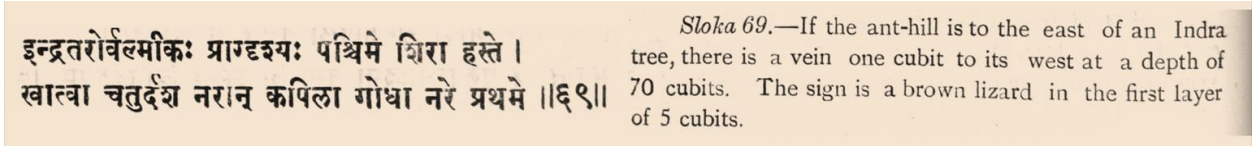
Many researchers contributed to the work presented in this chapter. All studies were performed in close collaboration with physiologist Prof. J. Scott Turner[1] and all experiments were conducted at his field site in Namibia 2011-12. The experimental methods were further developed with Dr. Nils Napp[2], Dr. Justin Werfel[2], and Prof. Radhika Nagpal[2,3]. The tool set for manual and semi-automated termite tracking was developed together with Justin Werfel and a Harvard undergraduate Erik Schluntz[3] respectively. Another student, Olena Bodila[4] spent countless of hours with the tracking software to provide comprehensive data sets for the analysis described in section 5.4; the analysis itself was done in conjunction with Dr. Paul Bardunias[1]. The 3D scanning tools described in section 5.5 were developed and tested with Nils Napp, and manufactured with Development Engineer Christian Ahler[2].

[1] Department of Environmental and Forest Biology, SUNY College of Environmental Science
[2] Wyss institute for biologically inspired engineering
[3] Harvard School of Engineering and Applied Sciences
[4] Harvard University Extension School

5.1 Introduction to Mound-Building Termites

Termites have existed for millions of years and are, in contrast to common belief, closer related to cockroaches and praying mantises, than ants [89]. Their impression on humans is traced back to ancient Egypt and India; in the 6th century the Indian scientist Varahamihira wrote that termite mounds were important indicators of groundwater and mineral deposits (Figure 5.1.a), a fact which has been supported many times since [90, 91]. However, the real fathers of termitology are Henry Smeathman and Herman Hagen who gave detailed accounts of termites in Africa and beyond [92, 93]. Many of the facts presented by Smeathman in the late 18th century are still considered to be true, and were supported by later research of both G. B. Haviland who first described *Macrotermes Natalensis* in 1898, and Sjösted on *M. Michaelseni* in 1914.

Termite colonies have a king and one or many queens. The queen is typically orders of magnitude larger than a worker, and immobilized by the large abdomen which produce about one egg every second. Depending on colony needs, the eggs are groomed to become reproductives (alates/nymphs), workers and soldiers (minor and major), respectively (Figure 5.1.b.c-d). Termites can live several years and molt several times, often morphing between castes. Minor workers are more prone to tend to queen and eggs, whereas major workers (Figure 5.1.c) make up the larger part of the construction force [94]. The gut flora of Macrotermes cannot digest wood; instead dead wood is foraged and packed into comb-structures for cultivating fungi deep in the nest.



इन्द्रतरोर्वल्मीकः प्राग्दृश्यः पश्चिमे शिरा हस्ते ।
खात्वा चतुर्दश नरान् कपिला गोधा नरे प्रथमे ॥६९॥

Sloka 69.—If the ant-hill is to the east of an Indra tree, there is a vein one cubit to its west at a depth of 70 cubits. The sign is a brown lizard in the first layer of 5 cubits.

Figure 5.1.a. Scan from book by 6th century Indian Scientist Varahamihira [90] explaining how termite mounds (then thought of as ant-hills) were indicators of water.

Figure 5.1.b. A. *Macrotermes Michaelseni* mound, less than 50 yards from a *M. Natalensis* mound (B.). C. Major soldier. D. Left to right: minor soldier, minor and major worker.

As described in section 1.1, a colony of *Macrotermes* houses up to two million individuals [95], mostly residing in the subterranean nest, underneath the mound. Most construction happens in foraging tunnels and in the mound, which is internally composed of a large interwoven network of tunnels all created from clay-like subsoil material. Termites turn over hundreds of kilograms of soil per hectare every year, making a huge impact on the local ecology [96-99].

Individually the termites are quite fragile, e.g. workers are blind and soldiers can neither eat nor drink on their own and must be fed. However, when part of their colony and the environment they shape for themselves, they are remarkably robust. Several sources have proposed the concept of superorganisms [21, 100-102]; that all individuals of a colony along with the nest and their mound can be viewed as one organism. In fact Turner has taken it one step further and proposed that the colony is strongly dependent on the mound not just for shelter, but to act as an artificial lung [103], to low pass filter turbulent winds on the surface to a low frequency 'breathing' in the center chimney, mixing nest and mound air.

How termites collectively achieve a specific mound shape and functionality is still unknown. In northern Namibia, two mound-building termite species thrive in the same environmental settings, yet build mounds of very different shapes; the *Macrotermes michaelseni* with mounds of conical bases crowned with tall slender towers and *M. natalensis* with much flatter mounds consisting just

of the conical base (Figure 5.1.b.a-b). Both species build closed chimney mounds. Interestingly, the two species of termites appear morphologically identical and are distinguished only by their mounds, and by close inspection of the major soldier caste. In my experimental work I focus on these two species; comparing reactions of both species when subjected to the same stimuli might shed light on what local behavioral choices make the global outcome of their efforts differ.

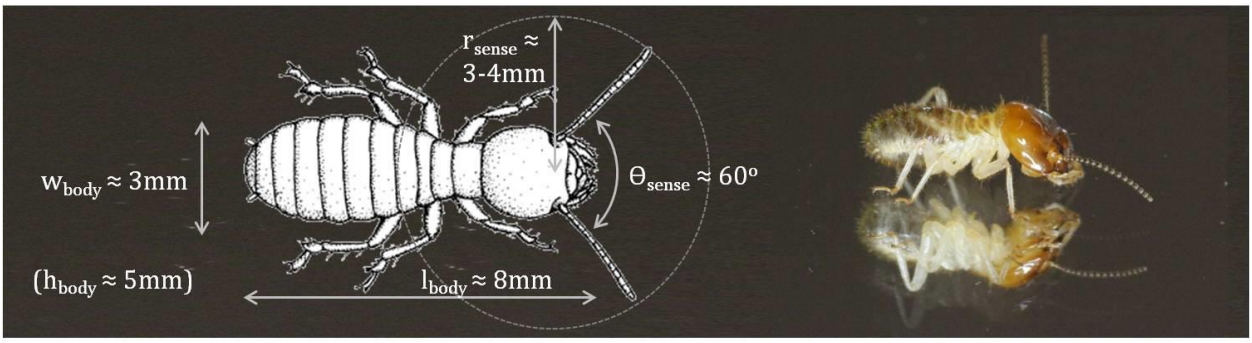


Figure 5.1.c. *Macrotermes Michaelseni* major worker on a mirror surface. The red/brown head has a hard shell, whereas the abdomen is soft and colored by the soil in the intestines. The illustration on the left shows approximate dimensions (the sketch is copied from Linsenmaiers, Insects of the World). Although termites can move their antennae almost 180^o, Lee et al. [104] have found that termites space their antennae out around 60^o while walking.

5.2 Related Work

This section describes traditional methods and tools used to study termites and other social insects, as well as research concerning behavior and stimuli related to termite construction.

5.2.1 In-Situ Experiments

Most experiments concerning termite construction behavior take place in confined laboratory settings ("ex-situ"), because setups in the mound are much more difficult. Insertion of instruments into a mound is a very invasive procedure (Figure 5.2.a.a) and causes disturbances that make it difficult to measure normal behavior. Instruments left in the mound are covered by soil in minutes.

Turner's group has published on a few in-situ experiments [105]. One experiment involved colored styrene beads which were pumped into the mound and nest at different heights. Assuming that termites manipulate beads as readily as normal soil, the mound was dissected after several weeks and the number of beads in different areas was taken to indicate how soil had been moved. Another experiment involved 'endo casting' on mounds (Figure 5.2.a.d), which is a destructive method of examining internal mound structure. While these experiments give us insights on the gross level outcomes of construction, they do not reveal much about the process by which the construction occurs. In section 5.6 I present a new method to observe repair directly in the mound.

Figure 5.2.a. Photos of *M. michaelseni* construction taken in Namibia 2011. A: J. Scott Turner and Eugene Marais attempting to decapitate a mound. The quote is from Maurice Maeterlinck in The Life of the White Ant, page 76 [106]. B: Termite construction in a Petri-dish arena. C: Spongy repair of a breach in a mound tunnel. D: Plaster cast of mound as part of work published in [105].

5.2.2 Ex-Situ Experiments

Traditionally, termite behavior is studied in confined experimental arenas, often Petri-dishes with reservoirs of soil, and recorded with overhead cameras (Figure 5.2.a.b). These experiments come with their own limitations, most importantly the fact that termite behavior deteriorates with the time spent away from the mound, lasting at most a day or two. Several research groups have used ex-situ experiments to study construction behavior and coordination; typically by observing termite paths, interactions, tactile stimuli, depositions, and excavations [105, 107-109]. Most researchers use manual inspection of video recordings or direct observations; only in a few cases has simple image processing software been used to provide estimates of how termites occupy space or move soil around an arena [105].

Manual tracking is slow, tedious, and prone to error. Automated computer vision methods may be worse at distinguishing subtle movements and changes in behavior, but allow fast collection of large datasets to improve the fidelity of future hypotheses. Automated tracking has already been put to great use in other fields of biology for example to explore group behavior of fish [110], mice [111], ants [112], and honeybees [113]. Ctrax [114] and SwisTrack [115] are popular open source programs for automated tracking of insects in closed arenas. These software packages have been used on fruit flies (for which Ctrax was originally intended), ants, cockroaches, and fish [113]. They rely on fixed background subtraction and constant velocity models, limiting their practical use in experiments where insect motion is more erratic and the background changes over time as is the case with termites engaged in construction. Other systems specialize in unmarked bees [113] and partially colored ants [46, 47]. One interesting recent example is the work by Mersch et al. [118], where they achieved perfect tracking of all individuals in an ant colony over several weeks using bar-code like tags that were physically attached to individual ants.

Trackers for termites differ in several aspects from those designed for other insects. Because termites have a limited life outside the mound, there is a strong time pressure on all experiments. It is difficult and time consuming to mark every single termite, therefore the tools developed in this thesis focus on unmarked termites. Also, most previous work is concerned with multiple-object tracking to study short-lived behavior, such as interactions, over relatively short spans of time (0.5-5min) [107, 113, 116]; however, to record a construction process which is comparatively slow, we need to analyze continuous tracks about three times that length (12-20min).

Individual insects often come close to one another, causing vision-based trackers to accidently split, loose, merge, or swap individual tracks. The software may perceive one individual as two, especially when the insects in question have segmented bodies. When insects remain motionless for a long time they start to be perceived as part of the background and become lost. Individuals are also often lost (the tracker becomes stuck on the background) when they exhibit sudden changes in

velocity. When insects crowd together it often leads to issues where several tracks are swapped, or one is lost entirely as two merge to one instead. In the experiments described here which seek to understand and distinguish the behavior of individuals over time it is especially important to avoid swapped individuals. Trackers for ants are especially prone to such errors because individuals climb on top of each other [107, 116, 119, 120]; fortunately termites are less likely to do so.

Although most tracking software makes use of the same overall methods, such as foreground extraction, Kalman filters and motion models, to deal with the issues mentioned above, it is still difficult to make these work in general settings and therefore it is not uncommon to design custom-made trackers for every new experimental setup. The tools presented in section 5.3 do not contribute new algorithms, but rather apply and customize existing methods to work with termites.

The methods mentioned above are 2D in nature and focused on tracking individuals, but not the changes they make to the environment. To the best of my knowledge, no work has been published on how to automatically and continuously quantify construction by social insects confined to experimental arenas. One difficulty is that the manipulated soil blends in with the background, and while it is very visible to the naked eye that can perceive depth, it is not easily visible in video recordings (even by a person). In section 5.5 I describe initial work on a tool set that uses 3D scanning to produce depth clouds at a level of resolution that allows identification of individual soil pellets as well as termites. This tool has the potential to dramatically change our ability to understand the process of insect construction.

5.2.3 Construction Stimuli

In 1959 Grassé introduced the term *stigmergy* [17] as one of the key elements in coordination of social insect behaviors. Stigmergy refers to the process by which social insects alter the environment to guide the decisions of future individuals; in other words it is a way for a large

swarm to pass information through a shared substrate, decreasing the need for explicit communication. Stigmergy can refer to both physical alteration of shape or to the addition of chemicals, such as pheromones, to the environment. Before long, researchers began to speculate that this hypothesis was too simple to explain all of the complicated structures in the mounds [121], (Figure 5.2.a). Stuart, for instance, pointed out in 1967 that negative reinforcement was essential to stop construction once started [122]. Recent algorithmic research have shown how global shapes can emerge through stigmergy [22, 59, 60], and subsequent projects focused on additional coordinating factors, such as templates, memory, wind, humidity, tactile interactions, and traffic flow patterns [61, 62, 63, 108, 109].

For his thesis in 1977, Bruinsma [94] carried out thorough experiments showing the properties of cement- and trail-pheromone, templates in queen chambers, and tactile information in *Macrotermes subhyalinus* (Rambur). My dissertation focus is on the development of tools to ease the study of the termites; however, in section 5.4, I present pilot studies regarding the role of cement-pheromone in *M. michaelseni* and *M. natalensis* termites.

5.2.4 Heterogeneous Workers and Division of Labor

Termites are strongly tied by social interactions; e.g. an isolated termite will quickly come to a stop and appear moribund. So far, inhibited by a lack of proper tools to track multiple individuals at once, most studies have focused on groups of workers and the combined outcome of their actions. However, that assumes all individuals are alike and operate with the same behavioral program. Conversely, nature has many examples of task specialization in social insects, both caused by polymorphism and polyethism [123-125]. It is possible that some workers, despite identical caste, are different than others; older termites, for instance, might have diminished sensory feedback. Difference in state dependent on recent experiences can cause differences in stimuli thresholds, and

willingness to mark trails as suggested for ants in [108]. Recent work in ants [107] and bees has also suggested the possibility of lazy workers, physically able, but unwilling to put in as much effort as their nest mates. In their work on groups of animals in motion, Iain Couzin's lab [126] suggests that uninformed individuals may stabilize the decision-making process by a few informed individuals. Likewise, it is hypothesized here that not all termites at a construction site acts the same. Tools to track and label the behavior of many individual termites in experimental arenas at once are presented in section 5.3. These have enabled exploratory studies described in section 5.4, the results of which indicate that division of labor may take place; specifically, where most termites focus on excavation, a few seem devoted to soil transport.

5.2.5 Species Differences

To the best of my knowledge only one paper has mentioned a pilot study on the difference between *M. michaelseni* and *M. natalensis* mounds. In [105] Turner, conducted experiments ex-situ with plugs of damp and wet soil. *Michaelseni* workers scattered their wet soil plug more widely than the damp soil, whereas *natalensis* workers displayed no difference. Upon repeating the test setup with plugs of freshly manipulated material and soil devoid of odor, *natalensis* workers showed preference for the former, whereas *michaelseni* workers were less biased. Turner suggest these results to be consistent with mound size, and that *natalensis* as the more avid stigmergic builder is more constrained to build smaller mounds, whereas *michaelseni* experience less inhibition resulting in larger spires. In section 5.4, I present further work to test the response of both species to freshly manipulated soil ("nest material") when compared to clean soil.

5.3 Ex-Situ Methods and Tools: Observing Termites in 2D

In this section I describe the main contribution of my work related to termites: methods and tools to record, track, and automatically label the behavior of individual termites engaged in collective construction in 2D experimental arenas in the lab. First, I describe methods for collecting termites and running ex-situ experiments. Second, I describe tools developed to allow reliable tracking of position and orientation of termites in the arenas, both manually and semi-automatically. Finally, I describe a tool developed to semi-automatically label the behavior of individual termites using their position and orientation. Using these methods and tools I have produced some initial hypotheses on differences in behavior related to cement-pheromone stimuli, differences between individuals, and differences between species; these are described in 5.4.

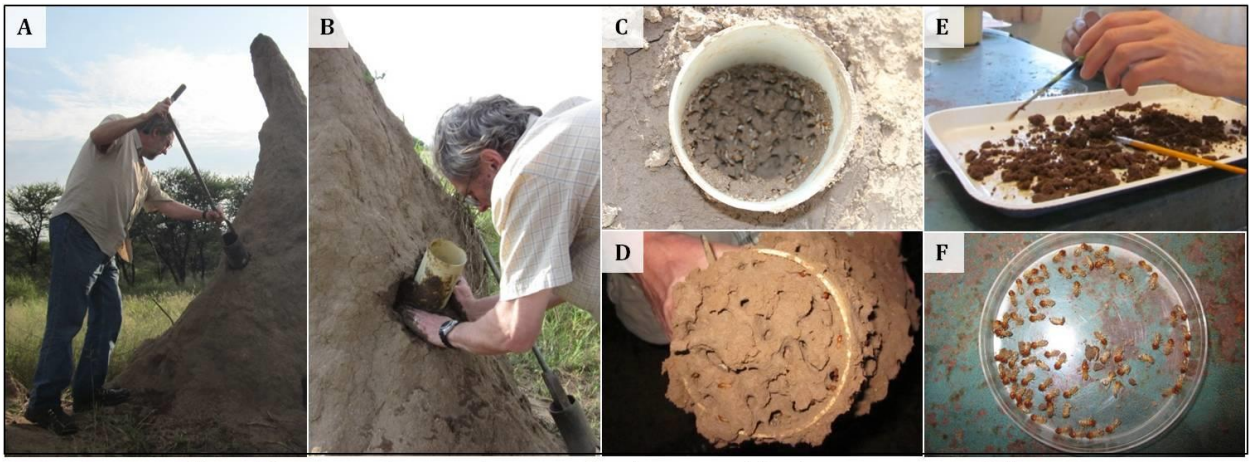
5.3.1 Method for Ex-Situ Experiments

Ex-situ experiments with mound-building termites are typically set up in 2D arenas with either plugs or a surface layer of soil [94, 105]. When introduced to a soil-filled arena, the workers start out in a confused 'acclimatization phase' with high activity milling about the arena, but eventually settle down and reengage construction. Digging and spurious deposits comes first, but after a while, specific sites become the focus of attention for construction as well. Here, I describe how to collect termites and soil, and how to set up the experimental arenas to test different stimuli separately.

Termites were collected from Omatjenne Research Station near Otjiwarongo, Namibia (-20.4°, 16.5°), the process of collection is described in Figure 5.3.a. As previously mentioned, termite behavior deteriorates with the time spent away from their mound; therefore, care was taken never to keep the termites in their containers for more than a couple of hours before experiments commenced. The experiments described in this dissertation all rely on termites gathered from *M.*

michaelseni or *M. natalensis* mounds, and are all performed with major workers, as these make up the largest part of the construction force. Minor workers might play a role in the construction process, but as advised by Turner, we focus on the efforts of the major worker caste first.

Besides termites, the experiments use 'clean soil' and 'nest material'. Clean soil refers to odor free soil gathered from termite mounds, left in buckets to wind and weather for a year, dried and then sifted to remove rocks greater than $1mm^2$. Nest material refers to freshly manipulated soil; a very moist clay-like substance gathered from the mounds, sorted for termites, and left in airtight containers as to not dry out. Nest material is used on the day of collection and only with termites from the same mound.



Termites were collected from mounds in northern Namibia (-20.4º, 16.5º). The process was as follows:

1. Dig a circular hole into a surface tunnel (A).
2. Pour ~1l of water into the hole to help termites initiate construction.
3. Mix mud and fasten PCV tube so that the opening lines up with the opening of the tunnel (B).
4. Add loose cover, (e.g. crumbled up plastic bag), to keep other insects out, and air turbulence to a minimum.
5. 4-5 hours later, remove cover to find (C), the tube is now filled with termites trying to plug the hole (D).
6. Bring everything to the lab and sort desired termite caste from mud using a brush (E).
7. Temporarily store termites in a Petri-dish with wetted filter paper for hydration (F).

Figure 5.3.a: Procedure of collecting termites for ex-situ experiments.

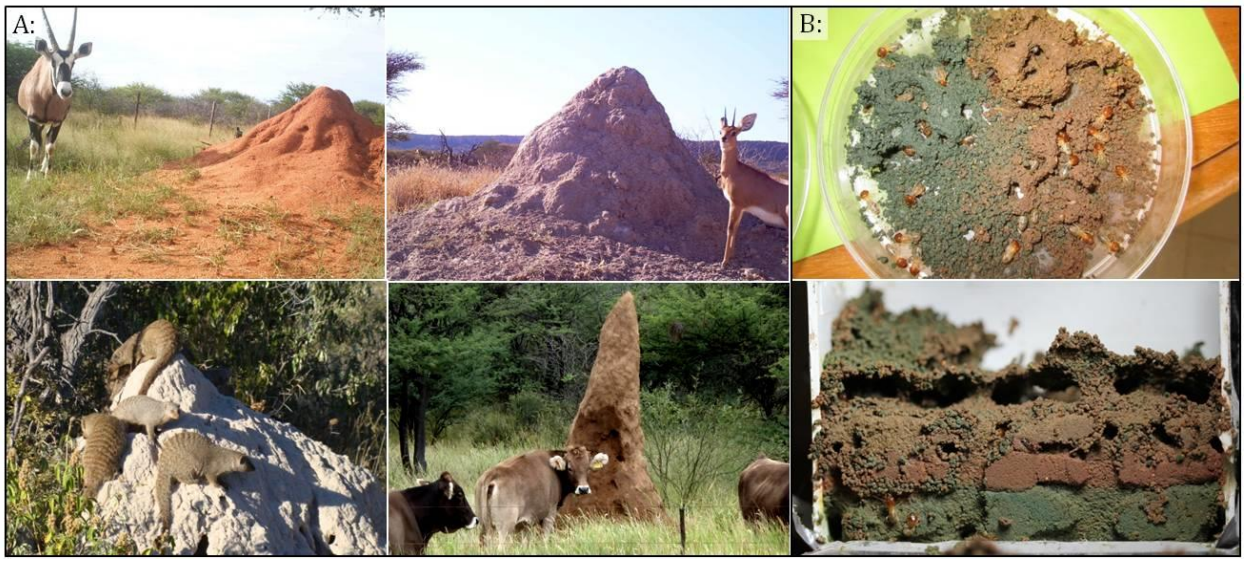


Figure 5.3.b. A: Examples of colors of soil in termite mounds near Omatjenne Research Station in Namibia. First two photos are used with permission from the Cheetah Conservation Fund, Namibia. B: Galleries built by termites in arenas filled with plugs/layers of clean and artificially colored soil.

On and around Omatjenne Research Station mounds are found on beds of grey, yellow, black, calcium-rich white, and red soil (Figure 5.3.b.a); i.e. termites appear to be indifferent to the type of soil they manipulate. Consequently, to work with visual trackers, experiments can make use of the color of soil on which unmarked termites appear most visually distinct. In the experiments described in section 5.4, red and grey soils were used mainly because they were the easiest to come by. Figure 5.3.b.b shows trials with natural and artificially colored soil; however, I found that termites can only be lured to build with the latter after being spurred on by the former.

The experimental arenas used in this dissertation are comprised of 87mm diameter Petri-dishes (Figure 5.3.c). These fit 25 termites comfortably; all other setups can be scaled area-wise around this number. To save time, effort, and keep termites unaffected, none of these experiments make use of marked termites. The floor of the arena is covered in a layer of soil to make tactile stimuli resemble that of the nest. Effort is made to make the surface of the soil consistent; small deviations

have been seen to bias the focus of excavation. The layer of soil is kept thin, 3-4mm, to prevent termites from digging under and out of sight. When only clean soil is used, it is typically hydrated to dry, medium, or wet moisture content: corresponding to 0.6g water/100g soil, 1.2g water/100g soil, or 1.8g water/100g soil. When used in the same experiment as nest material, the clean soil is hydrated to match the moisture content of the nest material. The moisture content of the nest material is found by drying out a small batch and weighing it before and after. A loose lid is added to protect from outside disturbances and to keep the termites in; covering it in a thin film of diluted detergent will keep it from fogging up during the experiments. The arenas are kept in moderate temperatures away from direct sunlight. As recommended by Turner, a small piece of wood was added to containers holding termites for a long period of time to provide comfort.

It was observed that termite construction behavior is generally biased towards the edge of the arena. This is problematic because it may cloud the results of any experiment set up to test specific stimulus, such as moisture content or pheromone. For practical reasons we decided to simply use symmetrical set ups where either side (stimulus-induced or not) is equally affected by the edge.

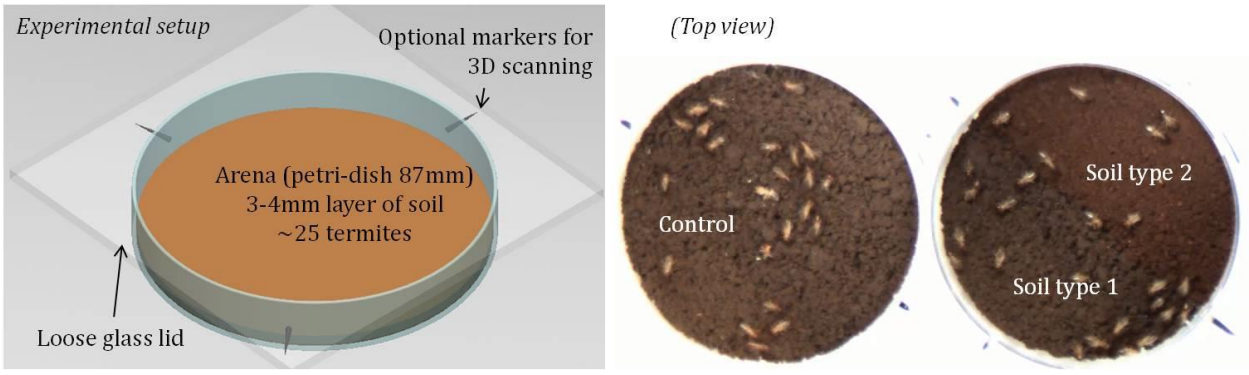


Figure 5.3.c. Typical experimental arena used in this dissertation.

5.3.2 Tools to Track Position and Orientation

Visual tracking of termites is complicated by the fact that they only work in crowds and preferably on soil where they immediately start constructing pillars and roofs to hide under (Figure 5.2.a.b). Recording termite paths, even during initial stages of construction in experimental arenas, however, may provide valuable insights on how termites coordinate construction. The following sections describe software to manually mark termite positions and orientations, semi-automatic scripts to track termite positions, and automated tools to detect orientation based on neural networks and steerable filters. These tools were initially developed to extract data from experimental recordings done in Namibia 2012 (Figure 5.3.d.a). Unfortunately, the setup was poorly implemented: it suffered from non-uniform lighting, too deep a layer of soil that allowed the termites to dig and disappear out of sight, and low quality recording with a frame rate of only 15fps and about 28pxl per termite body (Related work has reported settings with twice the frame rate and up to 1000pxl per individual [113, 116]). These issues can easily be minimized in future setups, but for now they are issues the software must deal with.

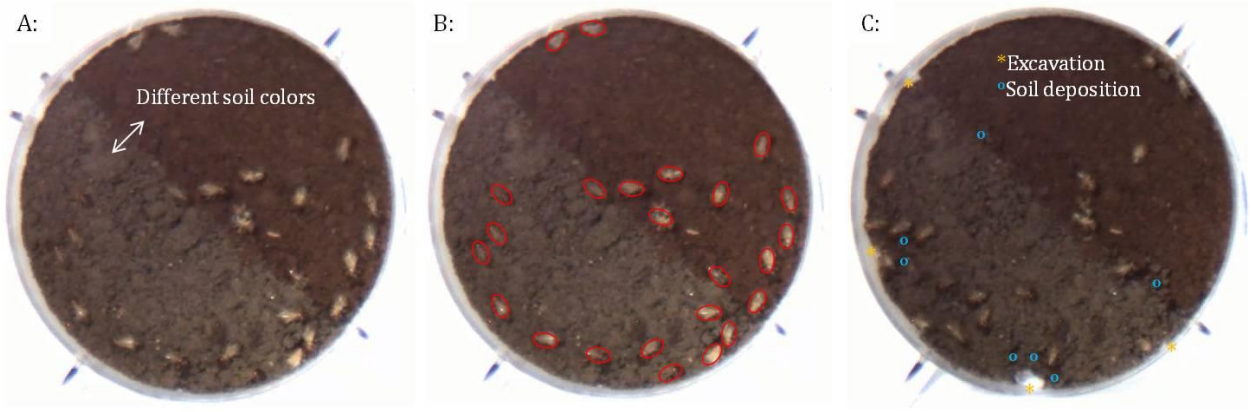


Figure 5.3.d. Experimental arena at 0min (A-B) and 13min (C). B highlights all the termites, some hardly visible to the naked eye. Notice issues with varying soil colors, non consistent lighting, poor resolution, changing backgrounds and termites digging down so far that only their rear is seen.

20 termites over 12min (5fps) were manually tracked using the tool described in section 5.3.2.1, and this data was used as a reference ("ground truth") to evaluate all tools developed. The semi-automated tracker of position takes user inputs and corrections in ambiguous situations. The automated trackers of orientation run without user inputs, but come with a Matlab [134] script that allows the user to browse through frames of the video afterwards to correct orientation errors. All auto- and semi-automated software is evaluated over the full 12min, and the average performance ±standard deviation is given for the first 8min (past 8min, some termite identities become ambiguous because they excavate to the point of visual occlusion). Orientation errors are intuitively divided into two categories; orientation estimates which are correct, but flipped 180^{o}, and the remainder. The first type of errors is fast to correct: the script allows the user to mark the initial and final frame between which the orientation is flipped and flip it back. The latter take longer to correct because the user must go through each frame and mark the proper orientation. The orientation-scripts are evaluated based on both types of errors.

5.3.2.1 Manual Tracking of Position and Orientation

A Matlab [134] script was developed to allow a user to mark anteriormost and posteriormost point on a termite body (head and "tail") in every frame of a recording (Figure 5.3.e). Videos were recorded at 15fps, however, for manual tracking it was deemed sufficient to only check 5fps. The script also allows manual recording of behaviors, divided into categories of moving, excavating, transporting soil, depositing soil, interacting and resting, described in detail in section 5.3.3. With this software a user only needs to focus on one termite at a time, making it relatively easy to deduce its position even in frames where it is somewhat occluded by soil and other termites. However, the cost is a very slow process: it takes around 80min to track a termite in a 13min video segment. 150 termites over 13min were tracked in this manner.

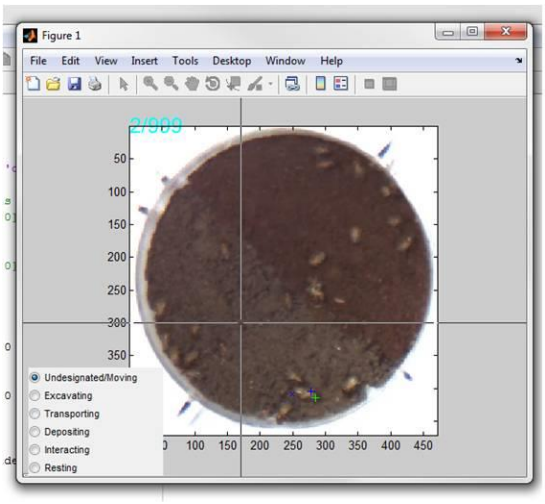


Figure 5.3.e. User interface of Matlab [134] script to manually mark head and "tail" position of a termite in every third frame of a video. Positions are recorded in pixels.

5.3.2.2 Semi-Automated Tracking of Position

A Matlab [134] script was developed to track the position of termites semi-automatically (Figure 5.3.f). It takes as input initial termite positions, and tracks based on visual detection and position estimates. Uncertain situations are automatically detected as large jumps in the position of termites in close proximity; the script then asks the user to verify the positions of the relevant termites.

Termite positions are estimated using the process shown in Figure 5.3.g. Foreground extraction is done in two steps. First, the background is estimated by the median of a set of 50 frames collected over 1000 frames and subtracted from the current frame. Second, to better distinguish termites, red and green color channels are added and the blue is subtracted. Image blob detection is done by applying a center-surround filter (also known as the Laplacian of a Gaussian) to reduce noisy pixels and prevent nearby termites from merging. The position estimate is a simple fixed size Gaussian centered on the position of the termite in the last frame. Finally, starting from the maximum value of their combined probabilities the script performs gradient ascent 2pxl at a time up to 10 times.

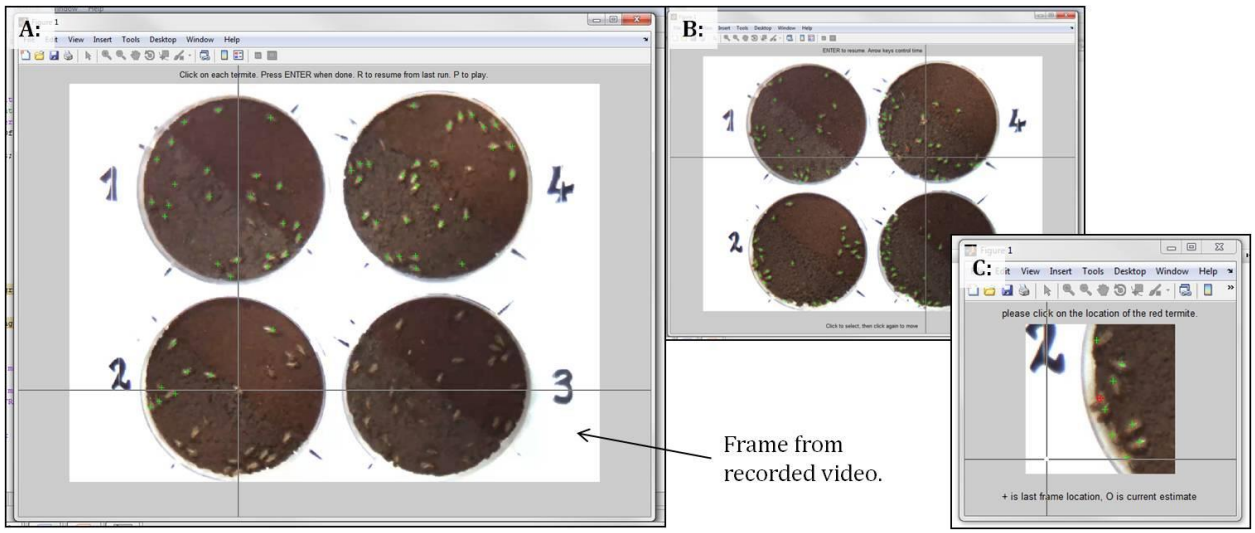


Figure 5.3.f. User interface of Matlab [134] script to semi-automatically mark termite body positions. A: The window to the far left shows initial user input marking the body of each termite in the first frame of the video. Next, the script automatically tracks termites from frame to frame, and asks for user intervention when data points appear uncertain (C). A user can also choose to step through the video one step at a time and correct particularly tricky segments (B).

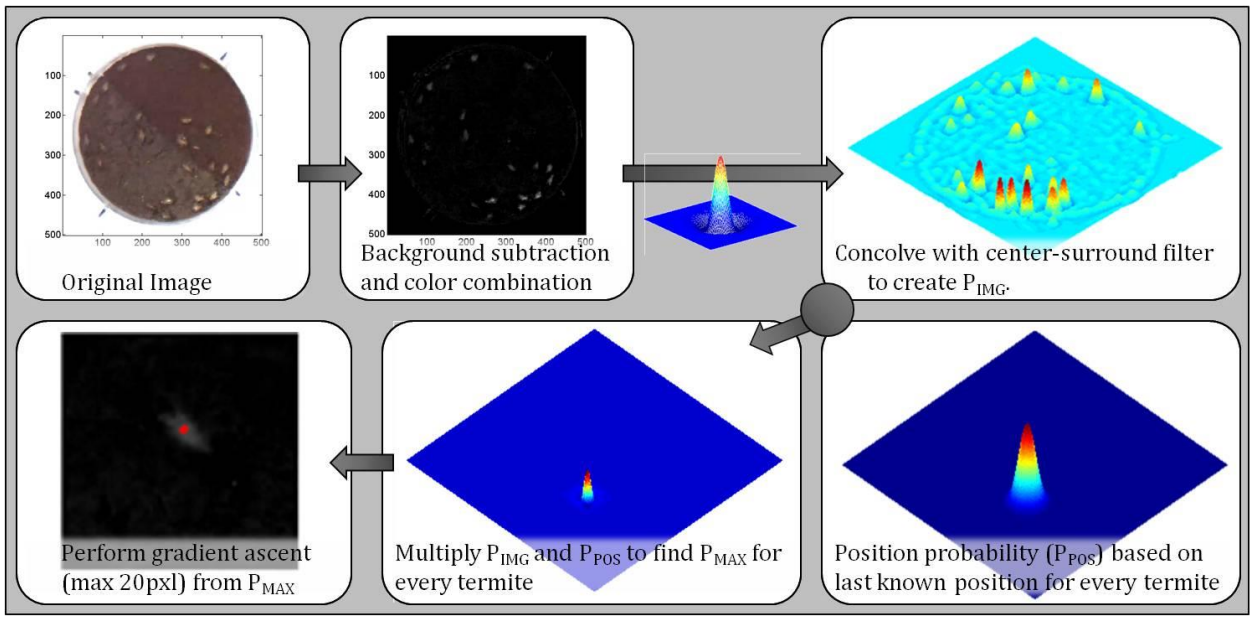


Figure 5.3.g. Procedure for tracking the position of every termite in a 2D arena.

Issues occur when termites occlude each other near excavation sites, move very fast, or temporarily when they dig underneath the surface layer of soil to reveal the white floor of the arena. Background subtraction will eventually reduce the errors caused by the latter. Non-moving termites will be negated out in the background estimate, however, because the script has a fixed amount of termites, their markers will remain in place unless another termite passes nearby. If termites are moribund/dead, it generally saves time not to mark them at all.

Results from this tracker were evaluated by comparison to manually marked positions of 20 termites over 12min (Figure 5.3.h). The average error from 0-8min is 2.91mm ± 0.97mm. The size of the average error is mostly due to the fact that the tracker marks the visually brightest point of the termite body (typically the center of the abdomen), whereas the manual observer marks the head position. From 8-12min 7 termite paths deviates. This is around the time when a lot of excavation happens and in 6 out of 7 cases termites were lost because they dug under the soil and it became ambiguous which termite reappeared on the surface. An observer using the manual software generally has a better chance of estimating which termite is which based on momentum and subtle visual cues in the video, whereas an observer using the semi-automated software has to keep track of about 100 termites at once (we recorded four replicas per video), and therefore will be more easily confused. Either way, future experimental setups should ensure that termites can never excavate so deep that they disappear entirely from the view of the camera. The last error (termite number 8) appears to be repeatedly lost and found. This happened because the termite had removed a piece of soil to reveal the white bottom of the arena and the tracker became stuck on the background rather than on the termite which continuously revisited the site. The error was too subtle for the observer to notice (her attention was on the mess of termites near the other excavation sites) and therefore these errors were not corrected. The termite was 'found' again whenever it returned to the same site of excavation.

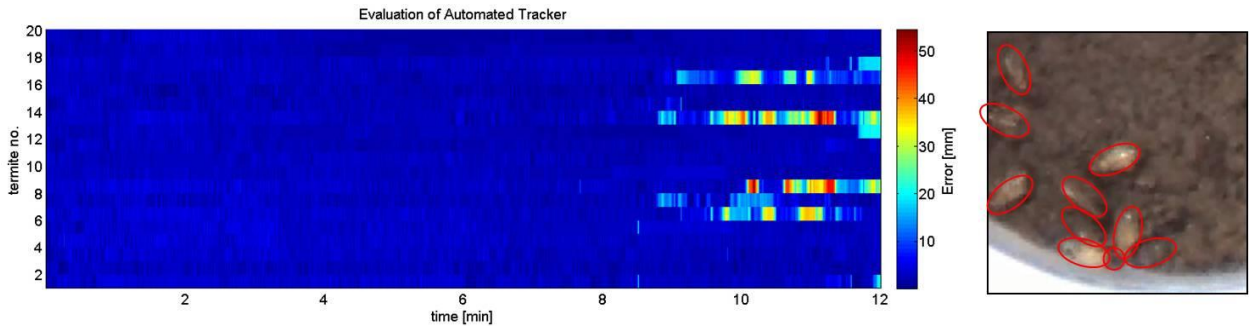


Figure 5.3.h. Errors in semi-automatically tracked positions when compared to manually tracked data for 20 termites over 12min. Picture to the right shows an example which can provoke an error: termites have dug so far down that they disappear from the view of the camera.

The semi-automated software allows one termite to be tracked in about a fifth of the time it would take to manually track it. This time is highly dependent on the quality of the video, and how much termites cluster. The software is easily adapted to new experimental setups simply by changing the parameters of the Gaussian filters applied to the image and position estimate.

The following sections propose automated software to also compute the orientation of the termites.

5.3.2.3 Automated Tracking of Orientation Using Neural Networks

Because most experiments conducted in this thesis used the same setup, and since one of those videos was already manually tracked, a neural network was trained to compute orientations for the rest of the recorded experiments. The neural network was composed of 900 input cells corresponding to a patch of 30pxl by 30pxl, a hidden layer of 100 cells, and 16 output cells corresponding to a resolution of 22.5^{o}, see Figure 5.3.i.a. The network was trained on 23 termites over 2:30min (55200 images). Regularization was used to favor simple models over non-simple

ones [127]. The output was filtered for spurious changes above 90° between frames and combined with the angle estimated from termite motion via a weighted average.

Running the neural network on 100 termites over 12min (15fps) took 4hrs. The automatically generated data set was compared to manually generated data for 20 termites over 12min (Figure 5.3.i.b). The average error from 0-8min is 23.11° ± 36.51°. When discounting errors due to 180° flips, the average error from 0-8min is 17.68° ± 24.02°. Most errors occur around crowded excavation sites when there is no motion and only the rear of the termite abdomen is visible. High resolution videos and better experimental setups will dramatically decrease the size and number of errors. As mentioned a small script was devised to let the user quickly step through the frames of a video to correct errors: on average this process takes 80min per arena of 25 termites.

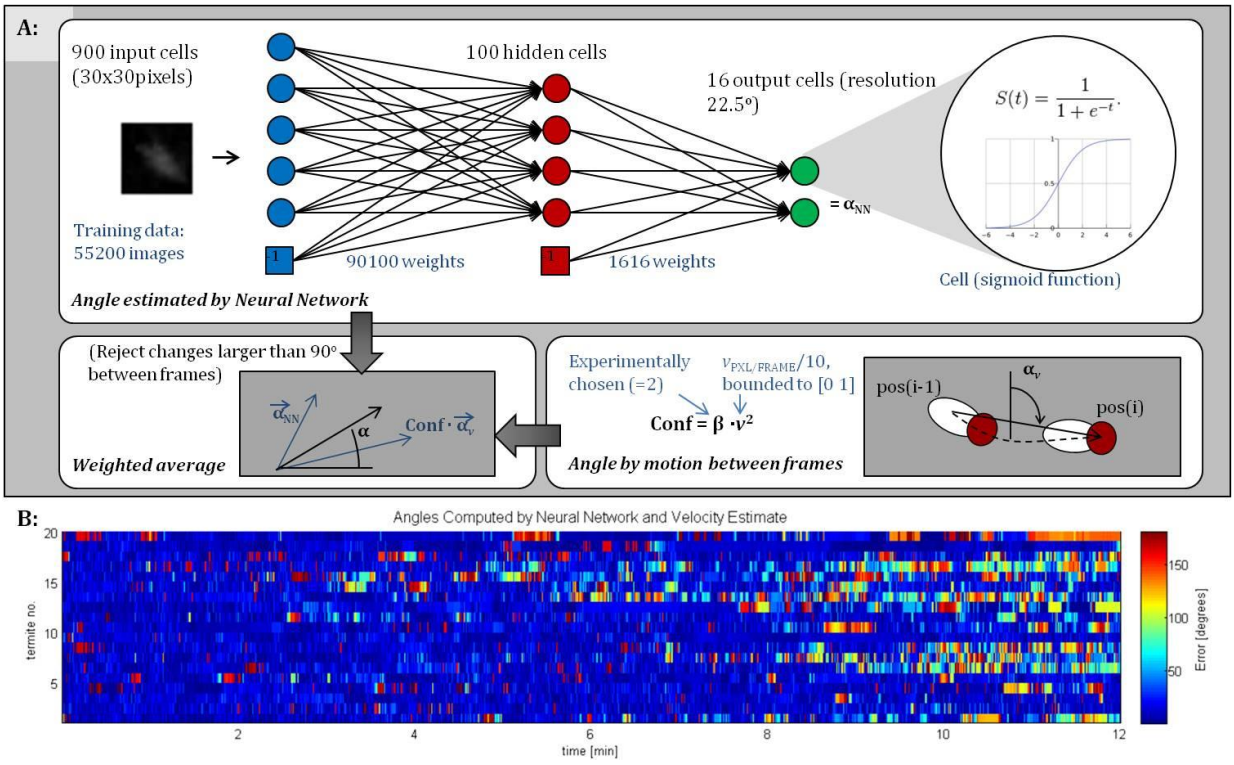


Figure 5.3.i. A: Neural network to estimate angles. The network output is filtered to reject changes above 90° between frames and combined, via a weighted average, with the angle derived from motion. B: The graph shows the difference between automatically and manually generated data.

5.4.2.4 Automated Tracking of Orientation Using Steerable Filters

The neural network works for the current set of experimental data; however, for every new experimental setup new training data must be obtained. To avoid this time consuming task, a script was devised to estimate angles based on steerable filters and termite motion (Figure 5.3.j.a). The steerable filters analyze change in brightness and compute the angle of the steepest change, Θ, corresponding to the horizontal axis of the termite across the abdomen.

Motion between frames is used to decide whether the termite orientation is within the interval from 0-180° (Θ_1) or 180-360° (Θ_2). This is done by computing the confidence, c_{frame}, as the bounded velocity per frames raised to the power of β:

$$c_{frame} = {v_{frame}}^{\beta} = \sqrt{\left(x_{frame} - x_{frame-1}\right)^2 + \left(y_{frame} - y_{frame-1}\right)^2}^{\beta}$$

Parameters were determined experimentally; β is 2, and the values between which the velocity is bounded are 0 and 10. Next, the confidence over segments of 200 frames (approximately 13s) is calculated as the sum of squared angular distance, multiplied by the confidence:

$$c_{segment} = \frac{\sum_{frame=1}^{200}(|\theta_1 - \theta_{VEL}|)^2 \times c_{frame} - \sum_{frame=1}^{200}(|\theta_2 - \theta_{VEL}|)^2 \times c_{frame}}{200\ frames}$$

When termites move very fast the corresponding segment have a high confidence, i.e., $|c_{segment}|$ is a high value, whereas a resting termite produces low confidence. The sign of $c_{segment}$ relates to whether Θ_1 or Θ_2 is the correct orientation. The reason for the large number of frames per segment is to ensure a reasonable number of samples with high speeds to get good angle estimates. Over a number of iterations (experimentally chosen to equal half the number of segments) the confidence of each segment is influenced by the value of its neighbors:

$$c_{segment}(n) = \frac{c_{segment}(n) + 0.5 \cdot \left(c_{segment}(n-1) + c_{segment}(n+1)\right)}{2}$$

The polarity of the final confidence in each segment determines if the orientation is recorded as Θ_1 or Θ_2.

The automatically generated data was compared to manually tracked data for 20 termites over 12min (Figure 5.3.j.b). The average error from 0-8min is 25.75° ± 44.94°; when discounting errors due to flipped orientation it is 15.06° ± 23.06°. Higher accuracy may be possible through a thorough investigation of optimal parameters. Again, a higher resolution video will also decrease the amount and size of errors; future versions might even use the color of the head to help prevent 180° errors.

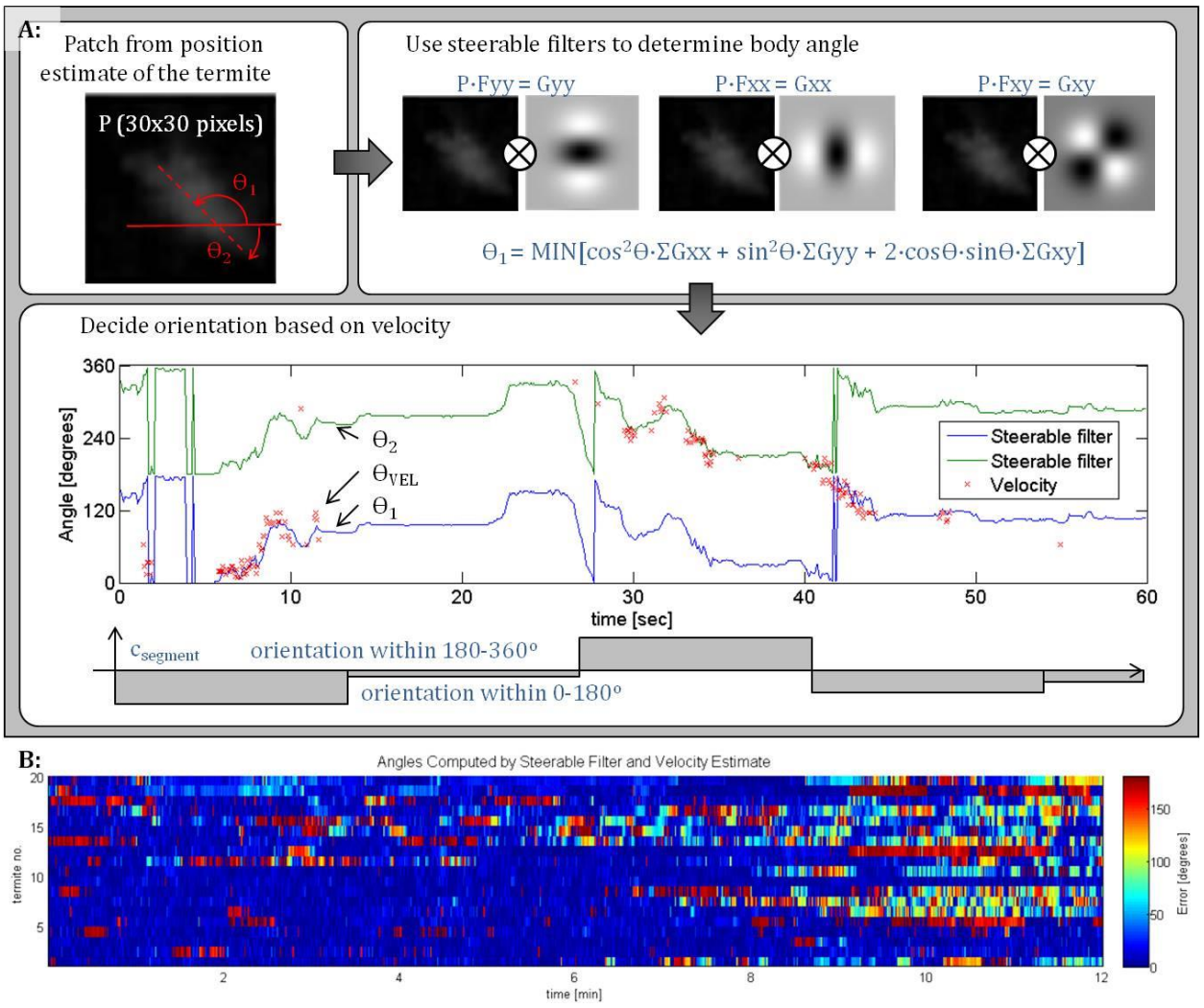


Figure 5.3.j. A: Steerable filters to detect body angle; velocity estimates decide heading (here, termite orientation follows the blue curve from 0-26s and from 39-60s, and the green curve from 26-39s). B: Error in automated angle detection when compared to manually labeled data for 20 termites over 12min.

As mentioned, the steerable filter tool has the advantage of working without the need for training data. However, if future setups record videos with higher resolution, the performance of the neural network tool would improve dramatically and it may then be beneficial to trade off time for training over time spent correcting errors. Because we are currently limited to relatively rough orientation estimates, the behavioral labeler described in the following section was developed to be minimally reliant on termite orientation. In the future, if orientation is more accurately classified, it will also improve the ability to automatically estimate behaviors that are more dependent on orientation, such as interaction.

5.3.3 Tool to Semi-automatically Assign Behavioral States

This section describes a Matlab [134] script devised to semi-automatically detect the behavior of termites based on their position and orientation, with the purpose of clarifying how much time termites spend in each behavioral state and why they transition between states.

Sites of excavation and their time of initiation are manually marked (Figure 5.3.k.a) as well as the time when a termite picks up soil and when deposition begins. Depositions are obvious because they are accompanied by cephalic rotation [128]. The behaviors are divided into 6 categories:

1. **Wander.** Behavior is labeled as wandering if individuals are not obviously engaged in another classifiable task.
2. **Excavation.** Behavior is labeled as excavating if the head of the termite is within 4mm of an excavation site (after its time of initiation) and the termite "sparsely moves". Sparsely moves is defined as a consecutive 8-frame window including the frame of interest, over which all of the following are true: a) the termite's head does not move more than 1.5mm from its position averaged over the last 5 frames, b) its head position does not change more than 2mm between

frames, and c) its body orientation does not change more than 20° between frames. Point a) deals with imprecise manual marking of termite positions; b) and c) ensures that there is no lag in the labels if the termite suddenly changes behavior. Here, excavation is a general term for the process that occurs close to an excavation site, including deposition when directly adjacent to the excavation site.

3. **Soil Transport.** Interval between when a termite picks up and starts to deposit soil (both endpoint events are manually detected).

4. **Soil Deposition.** Initiation of deposition is manually labeled; deposition behavior lasts a minimum of 2 frames, until either more than 20 frames has passed or the head of the termite moves more than 2mm from the drop point.

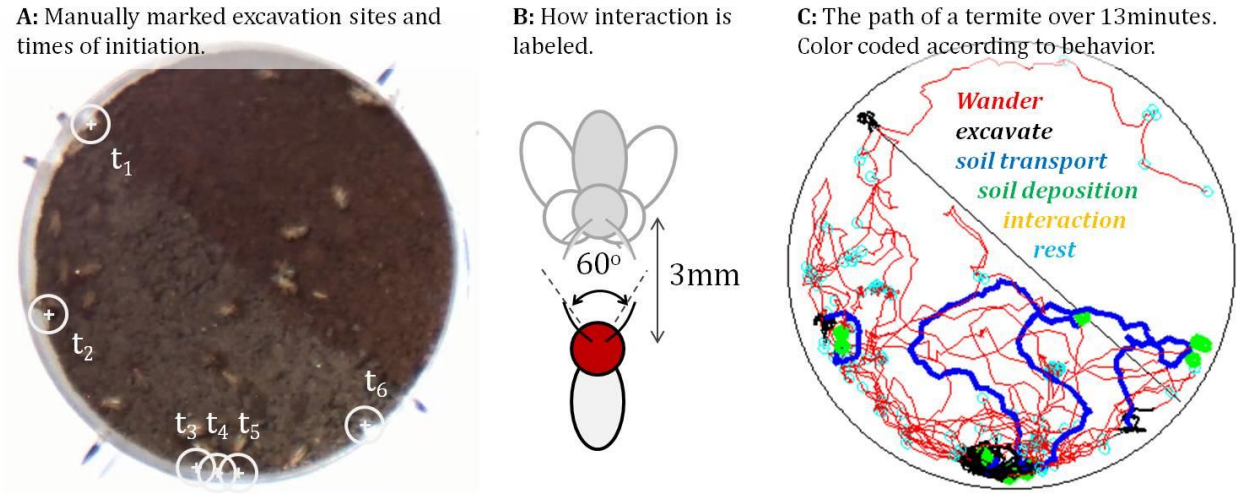


Figure 5.3.k. Parameters and output from the script to semi-automatically label behavior. A: shows how the center of excavation sites is marked along with their time of initiation. B: shows how interaction is labeled if the relative orientations of two termites differ by 150° to 210°. C: shows the path of a termite from the arena shown in A, color coded according to automatically labeled behavior.

5. **Interaction.** Termite behavior is labeled as interacting if two termite heads are within 3mm, their relative orientations differ by 150^{o} to 210^{o} (the angle formed by their antennae while walking is around 60^{o} [104]), see Figure 5.3.k.b, and both sparsely move.

6. **Rest.** Behavior is considered resting if a termite sparsely moved for a consecutive 12 frames including the frame of interest.

Behaviors are marked in the following order of precedence: transporting, depositing, excavating, interacting, resting, wandering, see Figure 5.3.k.c. Labeling the behavior of 25 termites over 13min in this manner takes about 5min.

The performance of the semi-automatically labeled data were evaluated based on manually labeled data for 10 termites over 13min. The software correctly labeled 90.93% ±3.71% of all behaviors, see Table 5.3.a. The most accurately labeled behaviors were wandering and excavating; the least accurate were interacting and depositing soil. Most erroneous labels occur when a termite transitions from one behavior to the next. Such boundary errors dominate behaviors in which termites spend little time, like interacting or depositing, but become negligible in behaviors that keep termites occupied for longer periods of time, like wandering or excavating.

Visual cues were helpful for detecting deposition and interaction; therefore these manual labels are more likely to be correct than automatically designated labels. However, because of the coarse quality of the video, cues were often ambiguous and inter-people agreement was low. Without an accurate notion, resting behavior was difficult to detect manually and when detected, the first few frames were often mislabeled as the behavior displayed prior to resting. As an example, more than 60% of the data that was automatically labeled as resting, but should not have been according to the manual labeler, was manually labeled as wandering. In both cases the semi-automatic labeler, as opposed to the manual labeler, is guaranteed to adhere to the designated classification of resting.

Table 5.3.a. Comparison of semi-automatically- and manually labeled behavior of 10 termites over more than 13min. "FalsePositive" are behaviors which were automatically labeled as the relevant behavior, but should not have been according to the manual labeler. "Correct" are identical labels. Transport behavior is manually determined and therefore omitted in this table.

Behavior Labels	**Correct** [% ±stddev = ($n_{CORRECT}$ / n_{MANUAL})]		**FalsePositive** [% = (n_{AUTO} - $n_{CORRECT}$) / (n_{TOTAL}-n_{MANUAL})]	
All	90.93 ±3.71	= (36336 / 39960)	-	
Wander	92.622 ±5.38	= (20073 / 21672)	8.11	= (21556-20073) / (39960-21672)
Excavate	90.90 ±3.27	= (9700 / 10671)	5.17	= (11214-9700) / (39960-10671)
Deposit	80.80 ±13.00	= (303 / 375)	0.11	= (346-303) / (39960-375)
Interact	67.39 ±5.89	= (31 / 46)	0.14	= (86-31) / (39960-46)
Rest	84.49 ±5.08	= (5017 / 5938)	1.54	= (5542-5017) / (39960-5938)

5.4 Ex-Situ Experiments: Nest Material vs. Clean Soil

Using the tools from section 5.3, I conducted a study to investigate cement-pheromone [17] which is thought to exist in saliva and recently manipulated soil ("nest material") and to play an important role in how construction is coordinated. I designed an experiment where termites were exposed to a surface layer of half nest material, half clean soil devoid of odor. I then recorded their behavior and used data from individual termites to examine several topics; including cement-pheromone as an inducer of deposition, differences in construction-related behavior between individual termites, and differences in construction-related behavior between two species of termites with different mound structures. Preliminary results of this study point to interesting hypotheses that traditional models for termite construction do not capture. The next step will be to conduct a more rigorous set of experiments to validate the ideas generated by this work.

Section 5.4.1 describes the experimental setup developed; 5.4.2 how nest material, when compared to clean soil, seems to have an arrestant property on termites which may have been confounded with cement-pheromone in previous studies; 5.4.3 show how soil transport of individual termites differ possibly indicating division of labor. In each of these experimental analyses, I compare the two species *M. michaelseni* and *M. natalensis*. Finally, in 5.4.4 I suggest method improvements and future studies. The work in section 5.4.2 was conducted with Dr. Paul Bardunias and is currently in submission with the Journal of Behavioral Processes.

5.4.1 Experimental setup

Termites of the caste *Macrotermes michaelseni* and *M. natalensis*, nest material, and clean soil were collected as described in section 4.5.1. 100 major workers of each caste were divided up into 4

Petri-dish arenas, about 25 per arena. The floor of the arena was covered in a 3-4mm layer of half clean soil, half nest material. The arenas were covered with a loose glass lid, and the termites were recorded with an overhead camera over 13min of experimentation (Figure 5.4.a). In most arenas, 2-3 individuals appeared moribund/dead, but the tracks of their nestmates showed no special interaction with them and their behavior was excluded from the data set.

All *M. michaelseni* were tracked manually using the tool described in section 5.3.2.1, all *M. natalensis* were semi-automatically tracked using the tools described in sections 5.3.2.2 and 5.3.2.3. The behaviors of all termites were labeled semi-automatically using the tool from section 5.3.3.

It is generally accepted that insects who are introduced to new arenas go through an acclimatization period. For termites, this period typically consists of rapid milling around the perimeter of the arena for the first 3-4min. To validate this, the number of termites wandering, as opposed to other behaviors, was examined over the full length of the experiment (Figure 5.4.b). After this initial period, increasing numbers of termites engaged in behaviors other than wandering around the dish. Consequently the following sections do not include the first 3min of data.

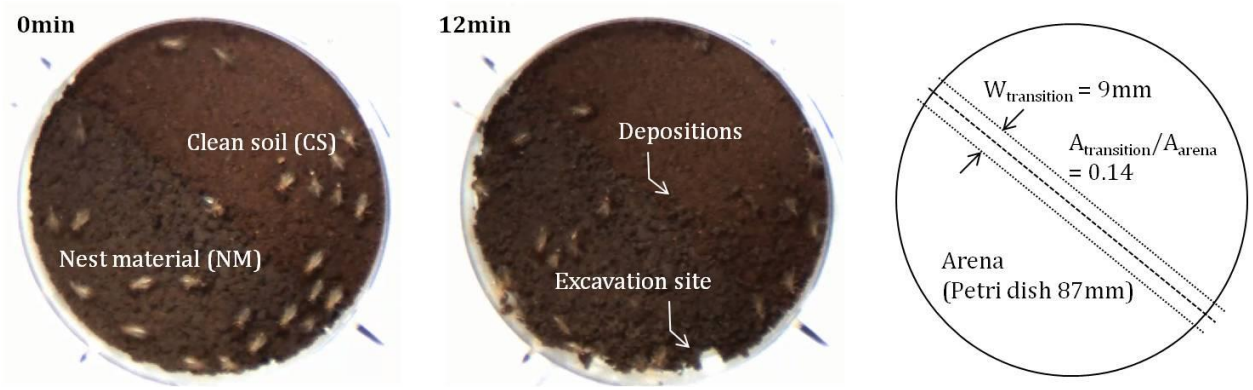


Figure 5.4.a. Experimental setup to test the effect of nest material (NM) versus clean soil (CS). Termite position and behavior was tracked over 12-13min, and studied based on residency on either half, or near the transition region between the halves. The transition region is defined as starting 4.5mm from the transition (approximately corresponding to termite sensing radius).

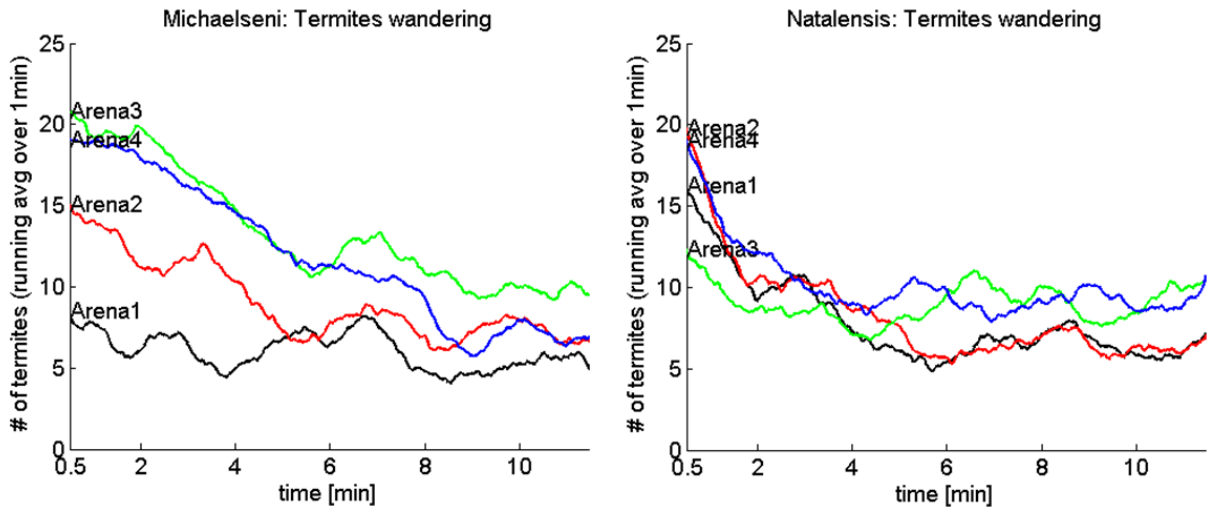


Figure 5.4.b: Number of termites exhibiting wandering behavior as a function of time, for each arena and species. Notice how the number stagnates after a few minutes of 'acclimatization phase'.

5.4.2 Arrestant Property of Nest Material

As mentioned in sections 2.4 and 5.2, models concerning termite construction typically exploit stigmergy for coordination; most assume that fresh deposits (nest material) are laced with saliva, the source of the putative cement-pheromone which acts as a chemical releaser causing nestmates to deposit probabilistically if present. Bardunias and Su [109], in their work on tunnel excavation in *Coptotermes formosanus*, more recently suggested that termites are directed to dig or deposit by traffic flow patterns and tactile interactions, and questioned the existence of this pheromone. Instead of a signal that specifically triggers deposition behavior, the odor of nest material may simply act to "arrest" termites; causing them to stay close by, but otherwise exhibit their normal behavioral patterns. Deposition is then more likely to occur in the presence of nest material simply because termites prefer a scent akin to a colony odor [129]. Chemicals that release aggregation behavior are known to exist in Blattoidea, which includes termites. While in most cases these

substances are not thought to be true pheromones [130], in the few that possess true aggregation pheromones, the substance is derived from salivary glands [131].

To estimate behavior preference, the time spent per behavior independent of residency was averaged over all termites in all arenas. To assess termite odor preference, the average time individuals spent on either substrate was compared via a paired *t test* at $\alpha = 0.05$ for each arena. For each arena we compared the average time individuals spent on either substrate engaged in different behaviors, via a paired *t test* at $\alpha = 0.05$. For clarity, the data from *M. michaelseni* is presented first, followed by the data from *M. natalensis*.

After the acclimatization period, *M. michaelseni* spent the most time wandering (40.41%±13.6), resting (28.67%±12.68), or excavating (25.78%±9.05), and less time transporting (3.75%±0.55), depositing (0.85%±0.47) and interacting (0.55%±0.69). Residency was significantly higher on the side with nest material rather than the side with clean soil (Table 5.4.a). Due to higher residency all behaviors conducted were significantly more likely to occur on the nest material (Figure 5.4.c).

Table 5.4.a. Preference of termites to reside on nest material over clean soil.

Arena	*M. michaelseni* Residency on NM	*M. michaelseni* Paired *t test*	*M. natalensis* Residency on NM	*M. natalensis* Paired *t test*
1	70.65% ±2.01%	n = 22, p= 1.44e-24	56.72% ±25.09%	n = 26, p = 8.10e-04
2	71.24% ±2.38%	n = 23, p = 6.25e-28	65.47% ±18.70%	n = 31, p = 4.91e-09
3	70.95% ±1.76%	n = 25, p = 7.01e-31	68.08% ±6.45%	n = 25, p = 1.81e-17
4	71.53% ±2.00%	n = 22, p = 4.48e-26	66.27% ±10.41%	n = 27, p = 2.15e-13

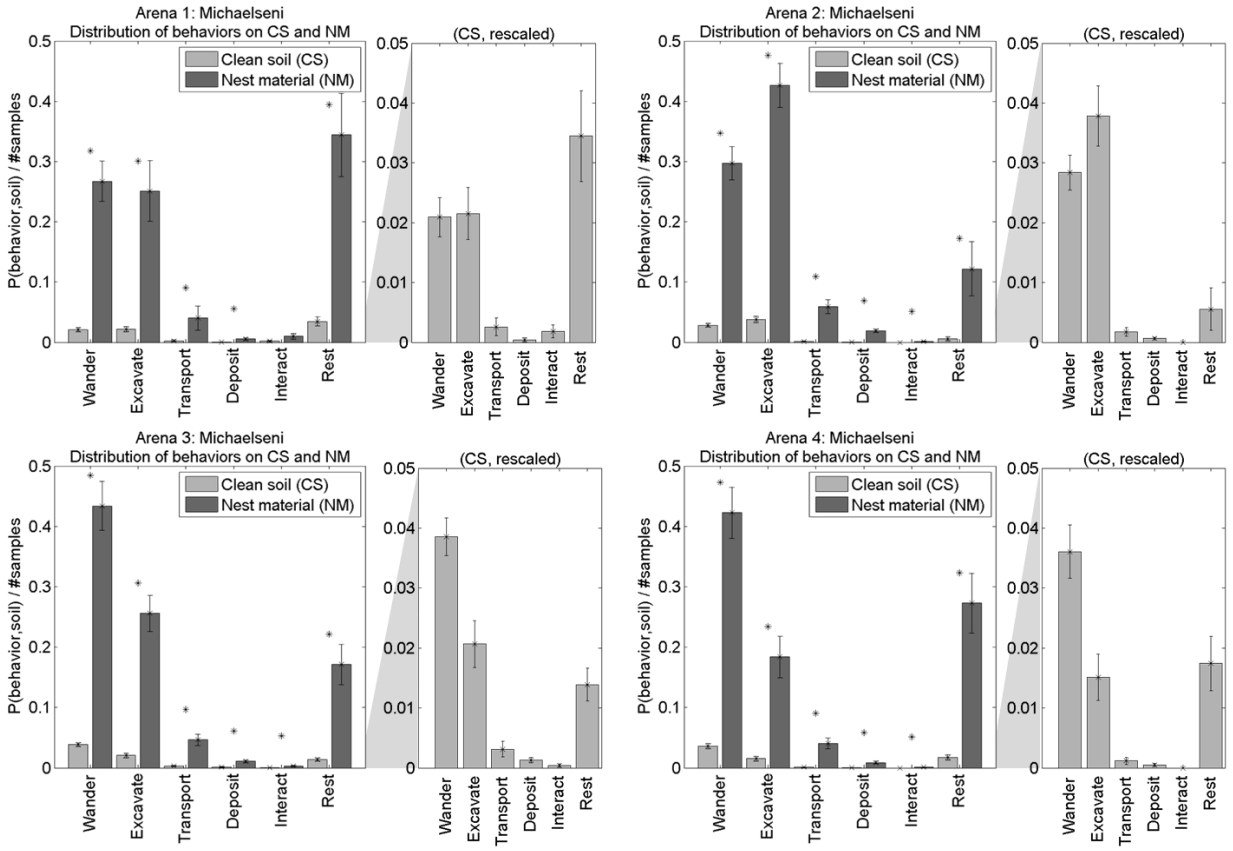


Figure 5.4.c. Data for *M. michaelseni*. Left hand graphs show the normalized distribution on clean soil (light) versus nest material (dark). Error bars show the standard deviation divided by the square root of the number of samples. Right hand graphs show an amplified view of the same distribution from clean soil for comparison with the distribution on nest material. * indicates significant difference as determined by a paired t test at $\alpha = 0.05$.

As with *M. michaelseni*, most *M. natalensis* workers were mostly engaged in wandering (39.75% ±10.37), excavating (26.36% ±14.87), or resting (26.34% ±17.00), and spent less time transporting soil (6.33% ±6.60), depositing soil (0.77% ±0.88), and interacting (0.44% ±0.46). Residency was in all but one case significantly higher on the side with a substrate of nest material rather than the side with clean soil (Table 5.4.a). Again, due to higher residency, behaviors conducted by termites were also significantly more likely to occur on the nest material (Figure 5.4.d).

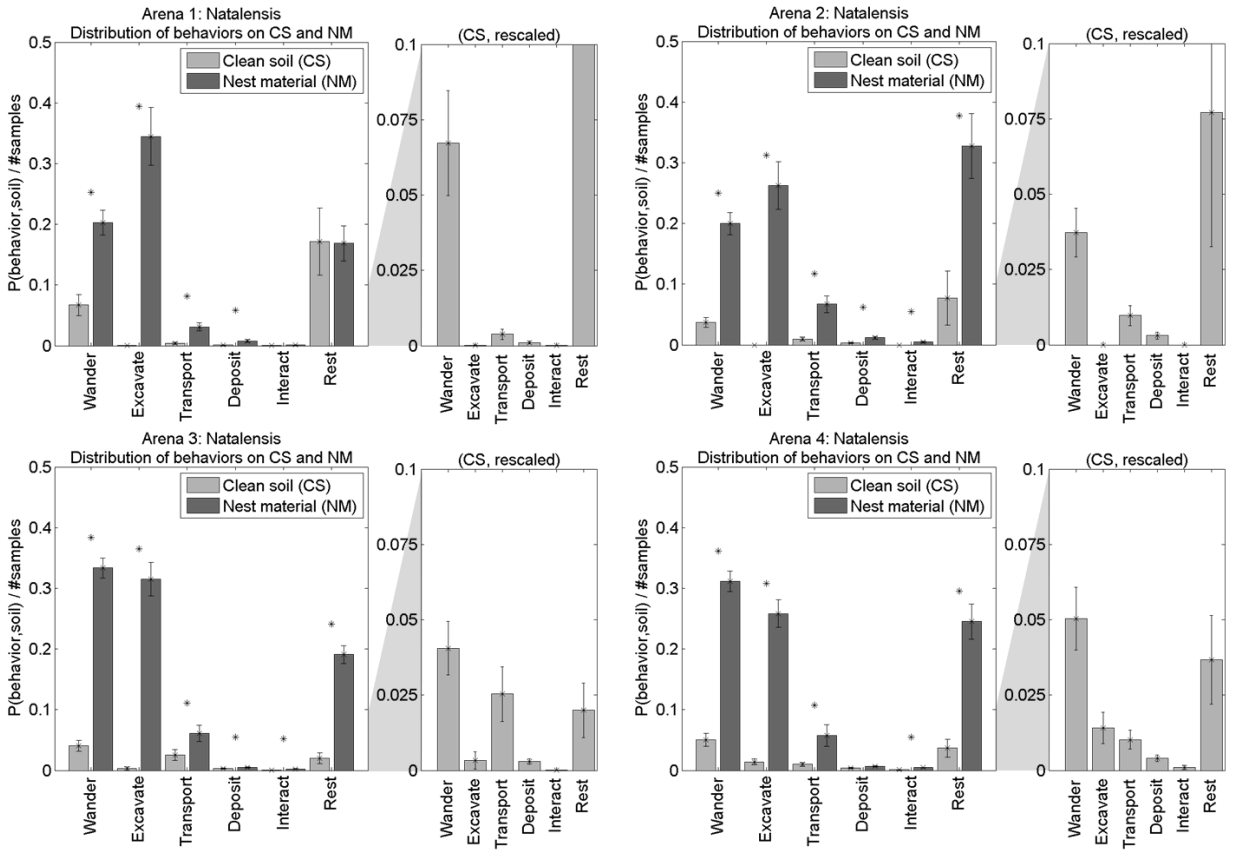


Figure 5.4.d. Data for *M. natalensis*. Left hand graphs show the normalized distribution on clean soil (light) versus nest material (dark). Error bars show the standard deviation divided by the square root of the number of samples. Right hand graphs show an amplified view of the same distribution from clean soil for comparison with the distribution on nest material. * indicates significant difference as determined by a paired t test at $\alpha = 0.05$.

It is interesting to notice that the distribution of behaviors on either type of soil appears similar for *M. michaelseni*, but not for *M. natalensis* (Figures 5.4.c-d). When controlled for residency, the total variation distance between the frequency of the behaviors on either type of soil for *M. michaelseni* is 0.02 (arena 1), 0.11 (arena 2), 0.06 (arena 3), 0.08 (arena 4), and for *M. natalensis* 0.37 (arena 1), 0.40 (arena 2), 0.34 (arena 3), 0.25 (arena 4). Furthermore, according to Table 5.4.a, *M. michaelseni* workers more consistently prefer the nest material than *M. natalensis* workers.

Thorough studies are needed to confirm and explain these phenomena. However, our findings indicate that nest material acts as an arrestant on both species when compared to clean soil; inducing not just deposition behavior, but an increase in all behaviors. This suggests a property of nest material that may have been confounded with cement-pheromone in earlier studies based on experimental arenas composed of clean soil. On a substrate of clean soil, the only nest material present will be soil excavated and subsequently deposited by termites. Additional deposition at these sites may simply occur because termites carrying soil are arrested in proximity to the deposited material. Termites in-situ rarely build on clean soil; most construction occurs as an extension of an existing nest. On a substrate of nest material, it seems likely that the odor of new depositions provides little or no signal above the background, and that other cues dominate. The arrestant property of nest material suggests a more subtle role for olfactory cues than a simple pheromone that induces deposition behavior.

5.4.3 Division of Labor and Soil Transport

Previous research has focused only on large collectives of termites and how they behave as a group; however, the semi-automatic tracker of position and orientation (section 5.3.2) enables studies of individual termites over long sequences of time. Here, I seek to determine if all termites act the same or if some exhibit different behavior, i.e. if division of labor takes place, based on their inclination to soil transport and their preference for the transition region between the two types of soil. In these exploratory experiments most soil was transported less than 1cm from where it was excavated, seemingly just to clear it out of the way to allow further excavation. Some soil was transported further, up to 6cm, often from the half with nest material to the half with clean soil; no instances of the opposite were recorded in either species.

The distribution of behaviors for every termite is shown in Figure 5.4.e, over the same set of experiments as in the previous section (4 arenas of 25 termites each for both species). Most termites exhibit the same behavioral pattern, however, a few termites are more actively engaged in construction (excavating, transporting, or depositing soil) than others who spend more time resting and wandering. A further distinction can be made between termites, which focus on excavation and transport soil only to clear out the excavation site, and those who specifically pick up soil to transport it further. Figure 5.4.f shows the amount of recorded depositions which were transported for more than 3s in a row; these graphs indicate that not all termites are equally interested in soil transport. In addition, though not striking, there appears to be some difference between species. Table 5.4.b shows that 30-40% of depositions carried for more than 3s by *M. natalensis* workers end up in the transition region between clean soil and nest material, despite the fact that this region only constitutes 14% of the entire arena. Figure 5.4.g shows paths of four *M. natalensis* workers in the same arena, two of which transport most of their deposits to the transition region.

Table 5.4.b. This table shows the number of recorded depositions, which were placed in the transition region ("TR") after being transported for more than 3s in a row.

Michaelseni	Deposits in TR	#Termites with more than 3 deposits in TR	*Natalensis*	Deposits in TR	#Termites with more than 3 deposits in TR
Arena 1	23.53%	2	Arena 1	41.18%	3
Arena 2	13.27%	2	Arena 2	31.67%	7
Arena 3	7.14%	0	Arena 3	40.00%	5
Arena 4	18.60%	1	Arena 4	29.52%	4

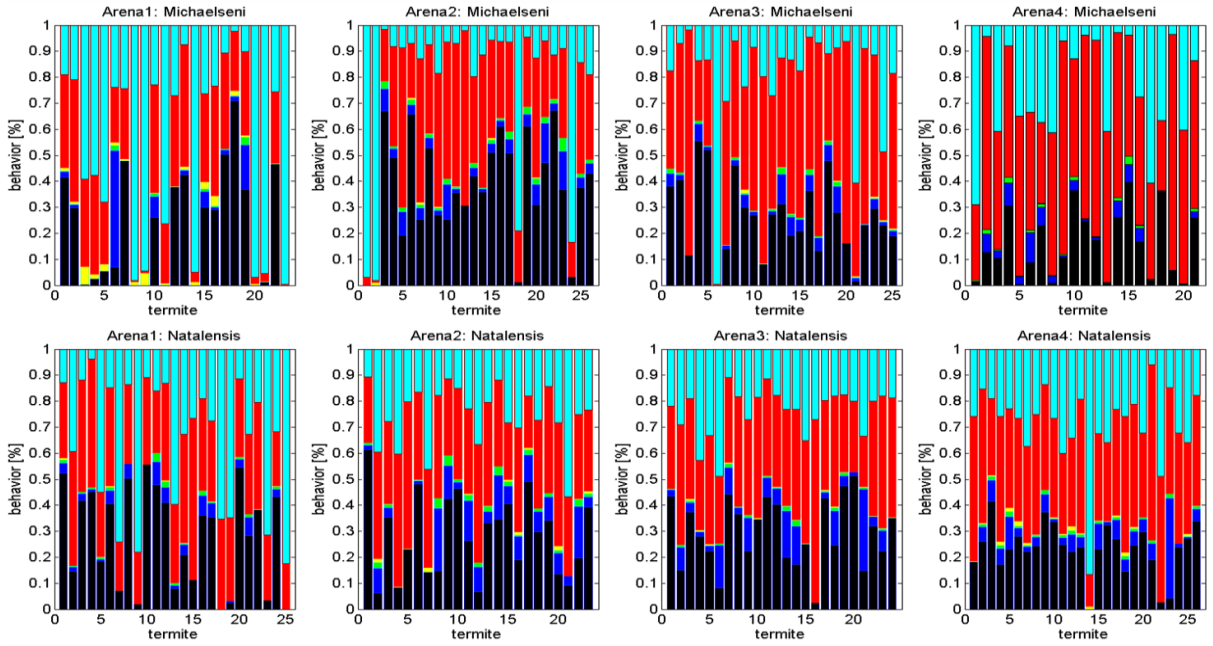


Figure 5.4.e. Distribution of behaviors for every termite; each column represents a single termite and the proportion of time they spent in each activity. Black marks excavation, blue soil transport, green soil deposition, yellow interaction, red wandering, and cyan marks resting.

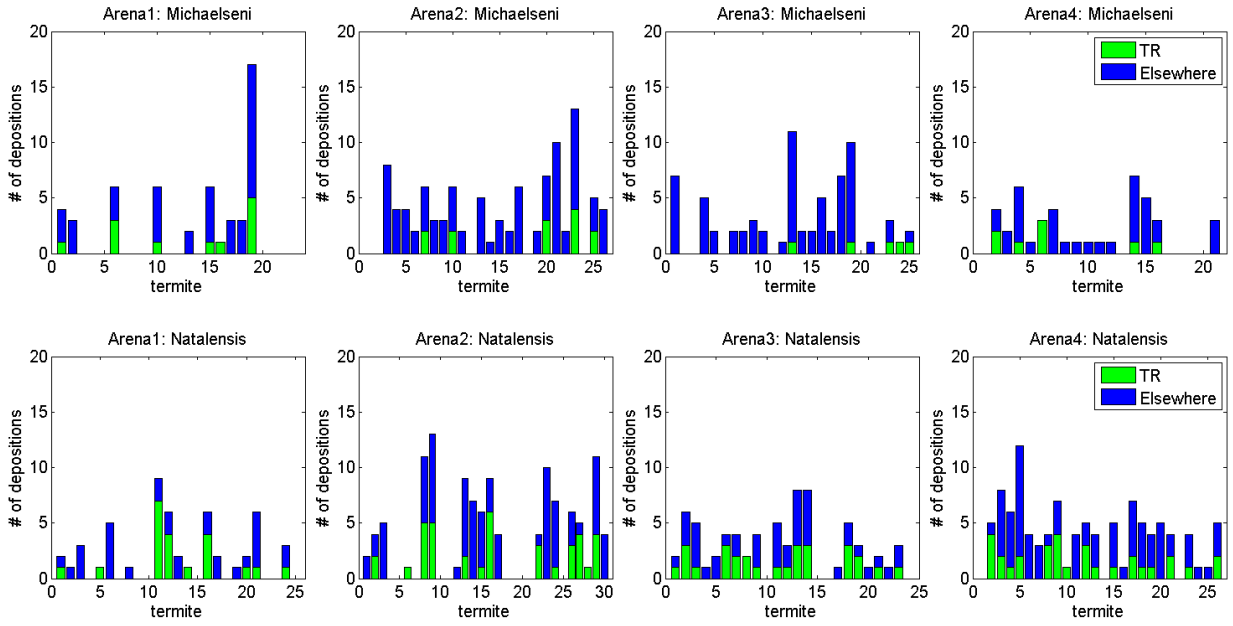


Figure 5.4.f. These graphs show the number of recorded depositions, which were placed in the transition region ("TR") or elsewhere, after being transported for more than 3s in a row.

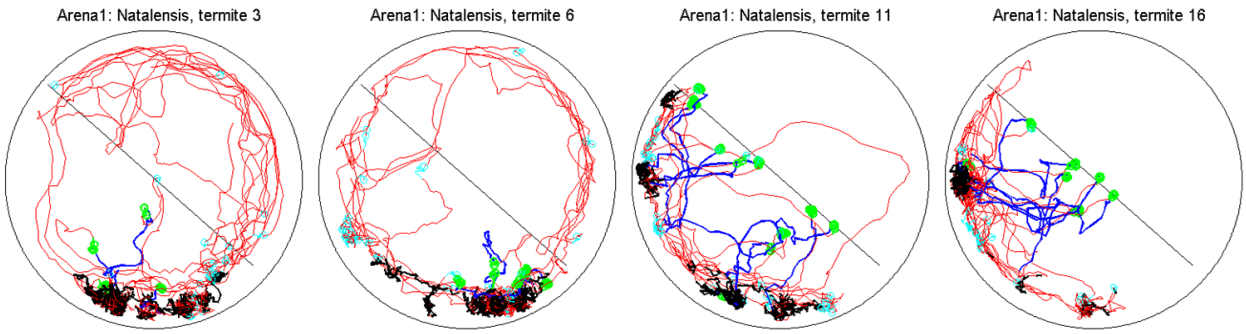


Figure 5.4.g: *M. natalensis* paths colored according to behavior: red marks wandering, black excavation, blue soil transport, green soil deposition, yellow interaction, and cyan marks resting. Notice how termites 11 and 16 repeatedly transport soil to the transition region, whereas termites 3 and 6 stick close by the excavation sites.

Clearly, this data set is too sparse to merit solid conclusions, but it does suggest interesting topics for further research. As mentioned in section 5.4.2 both species distinctly prefer to reside on nest material over clean soil, but as shown here soil deposition by *M. natalensis* workers seems more biased towards the boundary of the nest material. Intuitively, this may be explained by a desire to 'expand their nest'. In these collectives of 25 termites, most individuals are occupied with excavation and only a few remain dedicated to soil transport. Future research could examine what prompts termites to become 'transporters' and if the number of transporters scales with the size of the collective.

5.4.4 Future Work

The results of this section, and the process by which they were procured, has prompted several ideas and method improvements for future experiments.

To conduct a thorough analysis on species response to the putative cement-pheromone in fresh depositions, I suggest the following experiment. From each of 5 mounds, do 3 replicas with 100

termites per arena. Each arena should be comprised of large Petri-dishes (138mm diameter) and covered in a 2-3mm layer of nest material dried and moistened to medium wetness. Termites should be recorded with a high-resolution camera, no less than 15fps, over at least 13min, preferably longer. A large arena will decrease the likelihood of stimuli overload; the odor from depositions might be saturating the environment in a small arena. The thin layer of soil will ensure that termites cannot excavate under the surface and disappear from sight. Using only nest material emulates the real conditions of construction in the mound more accurately, ensuring that sites of construction does not emerge merely because termites are arrested in proximity to depositions on surroundings that otherwise lack familiar odor. High-resolution recordings will ease semi-automatic tracking. Using more termites and only a single type of soil will allow better data sets to categorize the distribution of behaviors. This may be necessary to understand the differences between *M. michaelseni* and *M. natalensis* which appear to be quite subtle.

The following experiment could complement the one described above to determine how termites integrate olfaction with tactile stigmergic cues. Prepare an arena like above, procure dried depositions from an old experiment and place it in the center of the arena. Although termites generally spend more time near the edge, if termites are strongly biased by tactile stimuli, it is hypothesized that columns will emerge near these depositions first. The same approach may be attempted with excavation sites.

Further experiments could test how termites of the different species are affected by water contents and other chemical properties of the soil. While we did conduct several experiments with different stimuli, the problems in methodology affected that data as well.

5.5 Ex-Situ Exploratory Tools: Observing Construction in 3D

To the extent of my knowledge no work has been published on how to automatically quantify construction by social insects confined to experimental arenas. Here, I present two tools to scan height maps of the arenas, with the purpose of collecting quantitative high-resolution data on how termites rearrange soil over time.

The first system uses structured light scanning [132] to record 3D height maps of soil in an arena before and after it has been manipulated by termites. While these initial and final scans give detailed geometric information about cumulative termite building activity over a fixed period, they do not provide information for construction dynamics within that period. The second is an ongoing effort to produce a height map which is dynamically updated while termites are active in the dish, using a laser scanner. We have tested both systems in the field, and preliminary evaluations are presented here; however, we have not yet had time to apply these tools to any rigorous studies. Both systems have the potential to be highly useful, allowing us to—to an increasing degree—automatically capture detailed geometric data on termite building behavior in controlled experimental settings. This work was published at a workshop at the International Conference on Patterns Recognition [26], 2012.

5.5.1 Structured Light Scanner

Structured light scanning is an inexpensive, but accurate, method of recording 3D height maps. Here, we use DAVID® Laser-scanner software [133] in conjunction with a Canon T2i camera and a small projector, iGo UP-2020. The process is illustrated in Figure 5.5.a: The projector emits a series of patterns on the target arena and each is recorded by the camera and processed in software to produce a point cloud. Each scan takes about 30s, and is repeated from three angles 120^{o} apart to

avoid occlusions. The three point clouds were automatically aligned in software using physical markers on the rim of the arena.

This setup achieved greater than 0.5mm resolution (soil pellets are typically 0.5-1mm diameter). After an initial scan, we added termites and let them behave for a fixed period (typically 30min), recording their actions with an overhead video camera. By the end of the experiment, the arenas were frozen to immobilize the termites. The termites were then carefully removed, and the final structure was scanned again. The final soil configuration would typically be within a total height difference of 8mm compared to the initial surface. This process allowed us to obtain geometric data like that seen in Figure 5.5.a.c-d. Finally the data was processed into a height map in Matlab [134]. The video data recorded were used for complementary tracking of individual termites using the methods described in section 5.3.2.

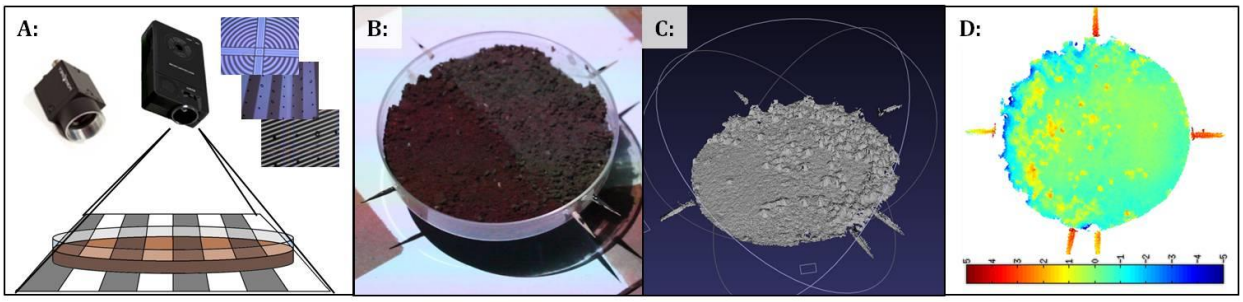


Figure 5.5.a. Structured light scanning of experimental arenas. A: Process with a camera recording a series of patterns projected on to the surface of the arena. B: Example of an arena being recorded; notice the physical markers on the rim of the arena to be used for automatic alignment in software between three scans. C: A single scan showed in Meshlab [135]. D: Three scans assembled in Matlab [134].

5.5.2 Laser Scanner

Continuous scanning was achieved through a multi-camera and line-laser setup as shown in Figure 5.5.b. In contrast to the structured-light approach described in section 5.5.1, which depends on a scene remaining static (i.e., without termites) during a time series of projected patterns, this setup gives instantaneous height information for a line of points illuminated by the laser. The laser illuminates a vertical plane passing through the center of the arena, and is rotated by a small angle between snapshots, giving a time series of updates like a radar sweep.

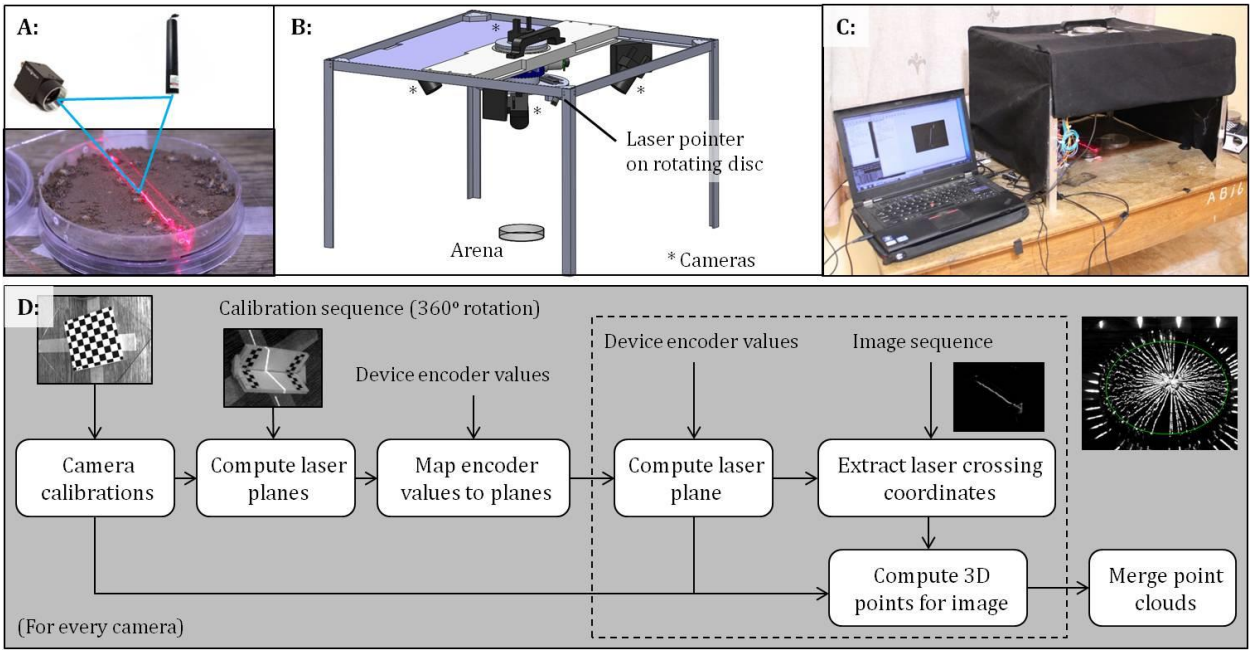


Figure 5.5.b. Laser scanner for continuous monitoring of termite construction in confined arenas. A: shows the geometric concept with a line laser and a camera. B: shows a simulated view of the device developed. C: Photo of the real device in Namibia 2012. D: Process by which the data from the device and images from the cameras on it are transformed into 3D height maps.

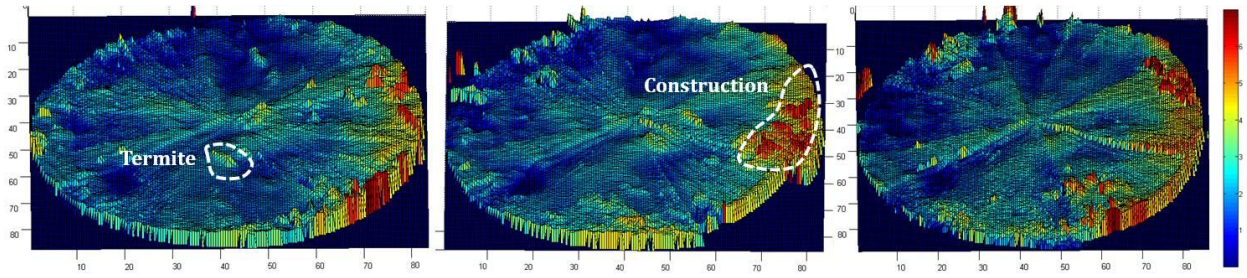


Figure 5.5.c. Data collected with the device shown in Figure 5.5.b. From left to right: height maps after 10, 20, and 30min of construction by 5 termites in a Petri-dish arena.

Three cameras (Point Grey®, FMVU-03MTM-CS) placed 120° apart take images to be used for 3D triangulation of illuminated points for each position of the line laser. The device holding the laser (AixiZ, AIX-650-5-1230) can rotate in steps as small as 0.2°, corresponding to better than 0.2mm resolution at the edge of a Petri-dish arena (87mm-diameter). A full 360° scan takes a little over 3min; initial results are shown in Figure 5.5.c. A fourth camera mounted directly above the arena records video data for position tracking. The laser is pulsed such that frames taken by the first three cameras with the laser on are interleaved with frames taken by the fourth with the laser off and strong ambient lights on. The video can be used for complementary tracking information as well as to identify the locations of termites, to disambiguate termites from soil in the height map. Because termites of the species of interest are blind, their behavior is not affected by the laser illumination. The entire scanning device can be broken out into modules and packed for travel convenience.

5.5.3 Future Work

The structured light scanner has given high resolution data, but is currently too slow to merit continuous observation while the termites construct. Investing in faster projectors and better

software (or developing custom software), has the potential to speed up this process drastically. This technique requires no moving parts, takes less than an hour to set up, and has a small form factor making it easy to transport in the field.

Conversely, the laser scanner obtains individual data points in the arena almost instantaneously and can scan the entire arena in about 3min, but needs better calibration and data treatment to distinguish individual soil pellets automatically (currently, only termites and construction can be told apart, see Figure 5.5.c). Reengineering the mechanical setup to spin the laser faster and modifying the software to deal with multiple lasers at once would speed up this process. Short term, the advantage of this technique is that we fully comprehend and control the software involved. However, compared to the structured light scanner, the laser scanner is large and heavy (even in the disassembled state) and requires mechanically moving parts which must be carefully calibrated with every setup.

Moving forward, I recommend investing time and money to make the structured light scanning technique able to capture data faster and to customize the software for full control. Furthermore, the process of turning the raw data into a useful data set, e.g. in Matlab [134], should be automated so that data validity can be easily checked while in the field.

5.6 In-Situ Exploratory Methods

Termites constantly monitor the mound and although no one has been able to prove so conclusively it is presumed that tunnels move, expand, and contract throughout the seasons. Non-invasive observation of the mound structure is difficult because permanently inserted probes are covered in soil and rendered useless. Instead, I here focus on *mound repair* with the purpose of developing a largely automated method to characterize colony response to mound breach and possibly further reveal differences between the two species of termites. The following sections describe issues with direct observation of mound repair and a new method developed to bias the termites to build out of their mound for easier observation.

5.6.1 Observing Mound Repair

Termites are not normally found in the mound, but when a breach occurs they appear within minutes, and soon the tunnel is swarming with soldiers and workers trying to plug the hole (Figure 5.6.a). Termites react both by direct stimulus and by a recruitment process [105]. Hours later, the tunnel will have many small construction sites each spurred on by a change in local stimuli, such as a wind eddy. Eventually, the attention is narrowed down to a few particular areas and then the breach is sealed; first sparsely by 'spongy build'; later by a solid plug. Afterwards, the tunnels undergo further modification to eventually become as smooth as the original.

Several experimental setups were tested for their ability to automatically collect data on the mound repair process (Figure 5.6.b). Unfortunately, none prevailed; probes inserted into a breach in the tunnel were covered in minutes, cameras recording from the surface of the breach recorded useless data because termites decided to plug the tunnel elsewhere.

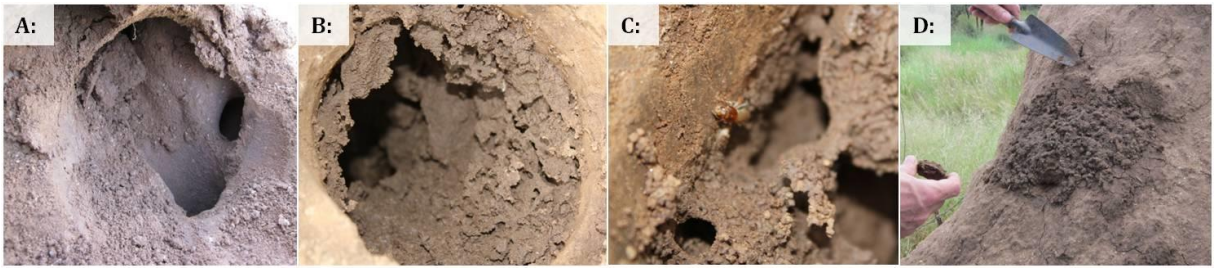


Figure 5.6.a. Various stages of repair in a *M. Michaelseni* mound. A: Tunnel network exposed just under the mound surface. B-C: 4-5hrs after the breach. D: Complete seal after about two days.

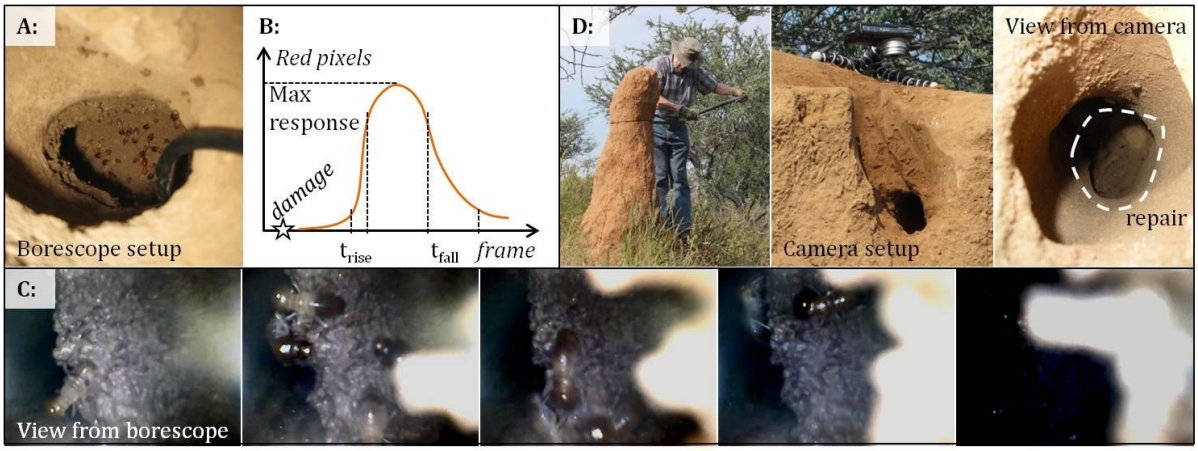


Figure 5.6.b. Failed methods to collect data on the mound repair process. A: Insertion of a borescope in the main tunnel leading to a breach. With a full circumference view of the tunnel measuring the amount of red pixels (termite heads) over time may reveal response characteristics (B). Within minutes of probe insertion termites repeatedly covered up the view (C). The second method (D) involves decapitation of a mound and camera setup to record the progress of the plug. Unfortunately termites decided to plug the tunnel far below the focal point of the camera.

5.6.2 Biasing Repairs

To avoid the need to record repair inside the mound, we devised a 'termite tower' to prompt construction outside the mound. Termite towers were inspired by ant farms and several experiments done ex-situ. They consist of two parallel glass plates joined by thin solid sides; one

end of the tower is placed in a reservoir of soil ex-situ, or along a tunnel in-situ, the other is covered by a cotton filter to let air in, but keep other insects out. Termites readily construct into the artificial environment, they are free to return to their nest, and the progress is easily recorded.

The result of an initial experiment, ex-situ, is shown in Figure 5.6.c. The soil covered area as a function of time is generated by imageJ [136] and an automated script developed in Matlab [134]. The activity is given by the derivative of the covered area, found by first smoothing the data with a tenth-order polynomial fit. The results were as expected, with high initial activity slowly diminishing after about 5min (presumably after cavities free of wind-eddies were constructed).

The ex-situ experiments encouraged tests with termite towers in-situ placed directly on the mounds (Figure 5.6.d). To guarantee steady light settings the experiments ran overnight. To prompt construction to proceed well into the tower, rather than a plug forming at the base, the glass sides were separated by 3cm (five times more than in the ex-situ experiment). After about 8hrs construction had largely ceased. The video data was processed like the data gathered ex-situ.

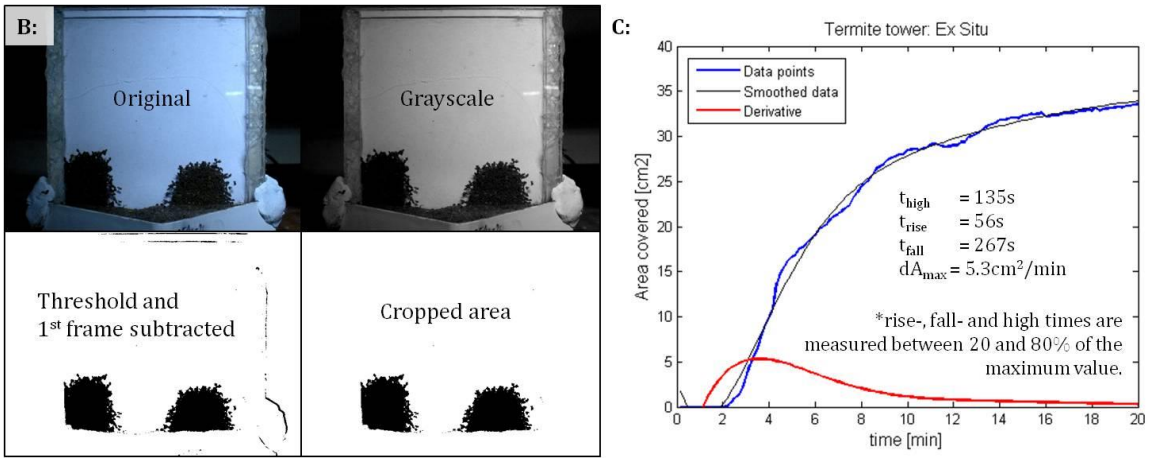


Figure 5.6.c. A: Snapshots from time lapse recording of termite tower ex-situ with 25 *M. michaelseni* major workers. B: Frame processed with ImageJ [136] by gray-scale, threshold, background subtraction, and masked to the area of interest. C: Graph generated in Matlab [134] showing construction progress as measured by black pixels in the processed frames.

Although the two experiments have very different experimental parameters, it is interesting to notice the difference in response time. The experiment ex-situ was characterized by a fast rise time and a slow fall time, whereas the experiment in-situ was characterized by a slower rise- than fall time. It is possible that the slow fall time of the experiments ex-situ is related to the termites being isolated from their nest, rather than their actual construction behavior. Conversely, the slow rise time in-situ could be explained by termites travelling further to get to the soil reservoir. Construction may have started deeper in the mound and not reached the tower until after the initial outburst of activity died down. Future trials may position termite towers closer to the nest, and possibly try towers of different widths to see how it affects the plug characteristics; e.g. density difference depending on distance from the surface.

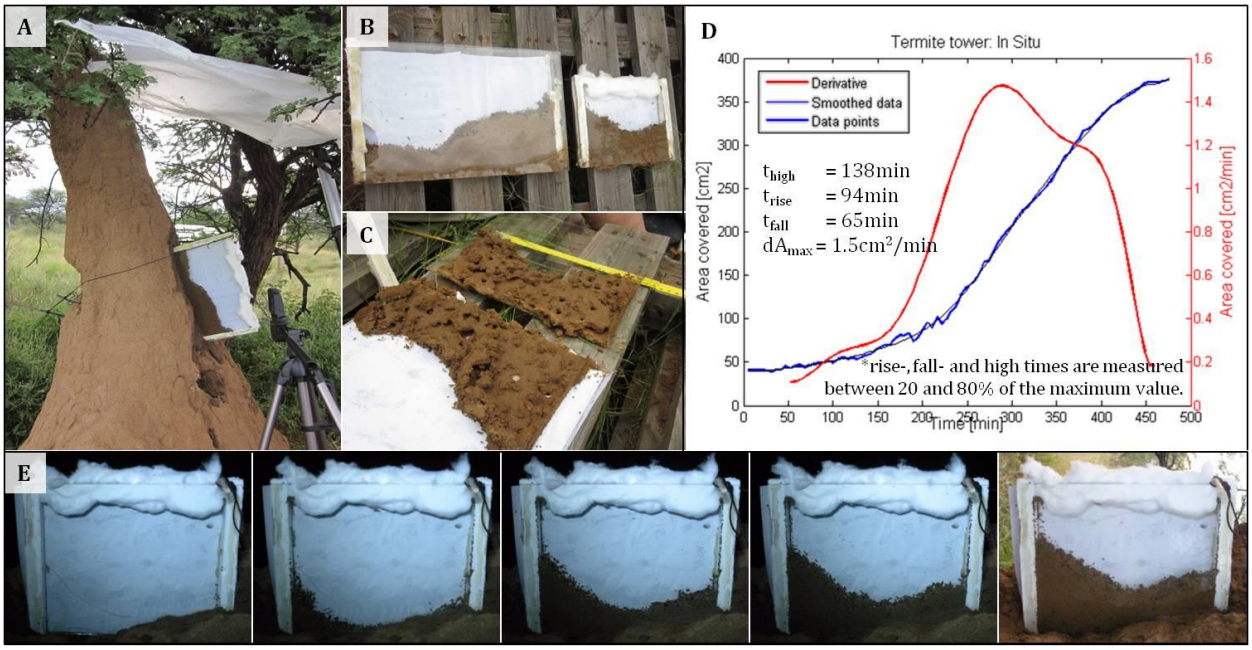


Figure 5.6.d: Termite towers in a *M. michaelseni* mound. A: Experimental setup. B: Construction in two towers; unfortunately, video was not procured for the larger tower. The smaller tower was 30x30x3cm³. C: Opened tower to reveal internal spongy build. D: Graph of construction progress. Video was treated as shown in Figure 5.6.c.b. E: Video snapshots, total length was about 8hrs.

Chapter 6. Conclusion

In this dissertation I have discussed the design of multi-robot systems for autonomous construction of structures larger than the collective building them. I am specifically interested in distributed systems because they can be efficient, multiple robots can work on the structure in parallel, and error tolerant, the progress of the collective is not dependent on a single robot. Starting with the simplest possible solution, I presented the TERMES system, an algorithmic framework and a robotic platform that needs neither central control, nor inter-agent communication to coordinate construction of user-specified 3D structures. Using these robots I have built structures more than eighteen times the volume of a robot, far larger than has been done by preexisting self-contained robots in the field of collective construction. I argued that the key to success is co-design, implementing passive mechanical features in bricks and robots to simplify control of the latter, and that reliability of the system is largely dominated by its ability to not just avoid, but more importantly recover, from errors. I then turned my attention to the mound-building termites that inspired TERMES; these are proof that large collectives can, in a very compliant manner, construct functional structures on scales much larger than the individuals without the need for central coordination. I developed a tool set to ease quantitative and qualitative data collection on termite construction behavior, and used it to perform exploratory studies and pinpoint future interesting research directions. My hope is that lessons learned from the study of the termites will make future robotic designs feasible additions to real world construction scenarios. Section 6.1 summarizes specific contributions; section 6.2 explains future interesting research directions.

6.1 Contributions

My main contribution is TERMES, a multi-robot system for autonomous collective construction of user-specified 3D structures.

We developed a high-level algorithmic framework to allow a collective of robots to construct a large class of structures without the need for centralized control or inter-agent communication, and proved that in a system free of errors the algorithm is guaranteed to lead to successful completion of the structure. We outlined the set of admissible structures, limited to structures free of overhangs with accessible paths throughout, changing no more than one brick height between adjacent stacks, and furthermore suggested algorithms to produce final structures without staircases.

I developed a robotic platform to implement the algorithmic framework in full, with three robots producing structures many times their own volume autonomously. The design of the platform was focused on a strong correlation between the design of bricks and robots, incorporating passive mechanical features into both, to ease control of the latter. A single actuator manipulator enabled robots to reliably pick up, transport and deposit bricks approximately their own volume, add bricks to the structure on top of other bricks, and extend the structure in the ground plane; an all-whegs design helped them climb one brick at a time; reliable navigation in multi-robot settings was accomplished with a total of 4 types of simple onboard sensors. A modular software architecture implemented mostly in finite state machines allowed easy modification of sub-routines as the hardware was reiterated.

As a secondary contribution, we developed new methods and tools for gathering quantitative and qualitative data on construction in two species of mound-building termites; including tools to semi-automatically track and label the behavior of individual termites confined to an experimental arena over long sequences of time, and exploratory tools to record 3D construction progress in

experimental arenas as well as exploratory methods to record the mound repair process in-situ. Using the software to track and label behavior we discovered that cement-pheromone may not play as important a role in coordination of construction as previously assumed; and that not all termites in the collective engage in the same type of construction behavior. We furthermore found indications that the two termite species react differently to recently manipulated soil, which could be a clue to why the mound-shape of the species differ.

6.2 Future Work

Near term, the TERMES system may improve by several additions. The algorithmic framework could incorporate error recovery and possibly add inter-agent communication to speed up construction. The hardware would benefit from the design of an automatic brick cache, a charging station where robots automatically recharge, and robots able to detach bricks and detect successful brick placements. Only single-path structures have been tested outside simulation; it would be informative to implement multi-path structures as well.

A longer term goal is to bring robots like these out of the lab; this requires consideration of how to ensure a smooth and level construction surface, how to navigate in changing light and noise conditions, and how to deal with external disturbances like grime and dust. The main principle behind the TERMES hardware is still amenable; robots can be made to navigate relative to the structure only and bricks can be shaped to simplify control of the robots.

Far term, bringing robots to assist at real world construction sites involves many challenges. Two major issues include system reliability, to enable longer construction sequences, and the set of admissible structures.

To improve system reliability the algorithmic framework should incorporate error recovery: how to deal with broken robots, misplaced material, brick imperfections, and external disturbances. Specialized robots could be employed to help remove broken robots from the structure. Writable markers on the brick could provide checkpoints to avoid navigational errors. Robots able to remove material from the structure could fix misplaced bricks. The algorithm could handle external disturbances by, e.g., relaxing the constraint on structure shape and have robots build around an obstacle.

To increase the set of admissible structures robots could be designed to handle overhangs for windows, archways, and roofs. Robots able to climb up straight walls would omit the need for

accessible pathways through the structure, i.e. eliminate the need for staircases. Robots able to handle real building material with proper attachment such as mortar, could produce sturdier structures. Robots able to construct with amorphous material could be used to deal with uneven ground surfaces or to incorporate preexisting environmental features into the buildings.

Finally, to better understand how to coordinate large swarms of agents in an effective and error tolerant manner, I recommend further studies of the mound-building termites. Specifically the methods and tools I have presented here enables a large set of studies concerning what stimulus induce and dissuade termite construction; how they coordinate construction of advanced outcomes like pillars and roofs; if, how, and why division of labor takes place. As mentioned in Chapter 1, I believe that viable autonomous construction by robots in the real world will be achieved through the combined research of abstract agents, physical robots, and their natural counterparts.

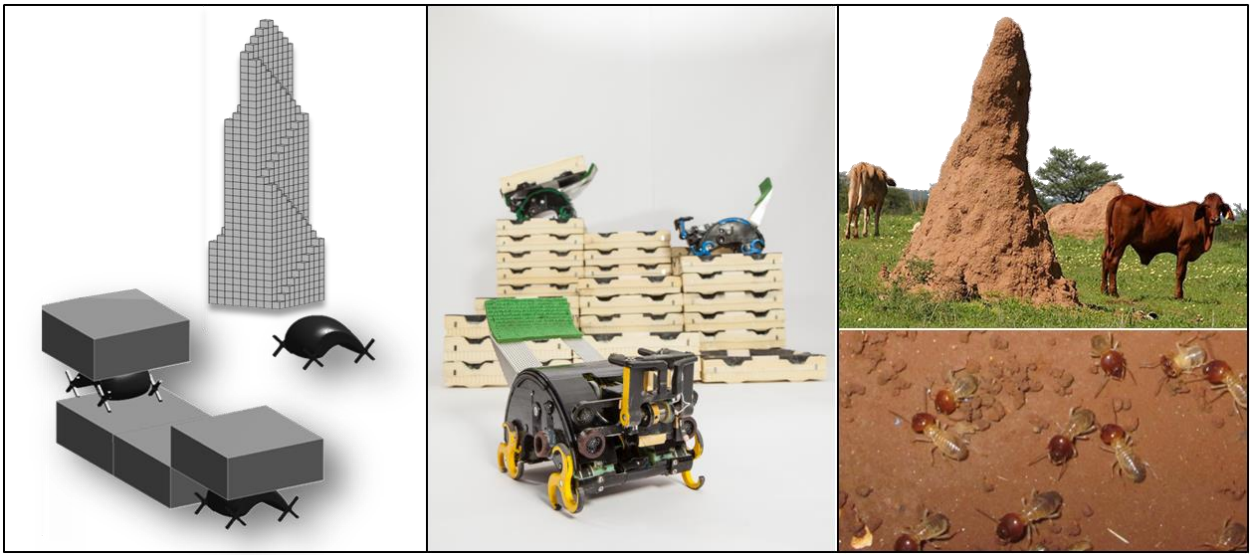


Figure 6.a. Left-to-right; abstract agents, physical robots, and their natural counterparts.

Bibliography

[1] Global Construction 2020: A global forecast for the construction industry over the next decade to 2020. Published by Global Construction Perspectives and Oxford Economics, UK, 2009.

[2] Occupational Safety and Health Administration website, United States Department of Labor, 2014. https://www.osha.gov/oshstats/commonstats.html

[3] Centers for Disease Control and Prevention website, National Institute for Occupational Safety and Health, 2014. http://www.cdc.gov/niosh/topics/construction/

[4] Bureau of Labor Statistics website, United States Department of Labor, 2014. http://www.bls.gov/iag/tgs/iag23.htm

[5] Khoshnevis, Behrokh. Automated Construction by Contour Crafting – Related Robotics and Information Technologies. Journal of Automation in Construction – special issue. The best of ISARC 2002, Vol. 13, Issue 1, pp 5-19, 2004.

[6] Buswell, R. A., R. Soar, A Gibb, and A. Thorpe. The Potential of Freeform Construction Processes. Proceedings of the 3rd International Conference on Innovation in Architecture, Engineering and Construction, pp 141-150, the Netherlands 2005.

[7] Joo, Hanjong et al. A study on the advantages on high-rise building construction which the application of Construction Robots take. Intl. Conference on Control, Automation and Systems, 2007.

[8] Khoshnevis, Behrokh. Automated Construction by Contour Crafting – Related Robotics and Information Technologies. Journal of Automation in Construction – special issue. The best of ISARC 2002, Vol. 13, Issue 1, pp 5-19, 2004.

[9] Willman, Jan, Frederico Augugliaro, Thomas Cadalbert, et al. Aerial Robotic Construction Towards a New Field of Architectural Research. Intl. Journal of Architectural Computing, Vol. 10, No. 3, pp: 439-459, 2012.

[10] Stroupe, Ashley, Avi Okon, Matthew Robinson, Terry Huntsberger, Hrand Aghazarian, and Eric Baumgartner. Sustainable cooperative robotic technologies for human and robotic outpost infrastructure construction and maintenance. Autonomous Robots 20, pp 113-123, 2006.

[11] Napp, Nils and Radhika Nagpal. Distributed Amorphous Ramp Construction in Unstructured Environments. Robotica, 2014.

[12] Yun, Seung-kook, Mac Schwager, and Daniela Rus. Coordinating Construction of Truss Structures using Distributed Equal-mass Partitioning. Robotics Research, Springer Tracts in Advanced Robotics, Vol. 70, pp 607-623, 2011.

[13] Worcester, James, and M. Ani Hsieh. Task Partitioning via Ant Colony Optimization for Distributed Assembly. Swarm Intelligence, Lecture Notes in Computer Science, Vol. 7461, pp: 145-155, 2012.

[14] Kelly, Jonathan and Hong Zhang. Combinatorial Optimization of Rule-Based Planar Distributed Assembly. In: Proceedings of 2006 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, pp: 3728-3734, Beijing, China, 2006.

[15] Støy, Kasper and Radhika Nagpal. Self-Reconfiguration Using Directed Growth. Intl. Symposium on Distributed Autonomous Robotic Systems, France, 2004.

[16] Howse, P. E. Termites: a study in social behavior. London: Hutchinson, 1970.

[17] Grassé, P.P. "La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et cubitermes sp. La theorie de la stigmergie: essai d'interpretation des termites constructeurs." Insectes Sociaux 6: pp 41-83, 1959.

[18] Turner, J. S. and R. C. Soar, 2008. Beyond biomimicry: What termites can tell us about realizing the living building. International Conference on Industrialized, Intelligent Construction.

[19] Napp, N. and R. Nagpal, 2014. Robotic Construction of Arbitrary Shapes with Amorphous Materials. International Conference on Robotics and Automation.

[20] Grushin, Alexander and James A. Reggia. Automated design of distributed control rules for the self-assembly of prespecified artificial structures. Robotics and Autonomous Systems 56, pp 334-359, 2008.

[21] Turner, J. Scott. The Extended Organism: The Physiology of Animal-Built Structures. Harvard University Press, 2000.

[22] Theraulaz, Guy, Eric Bonabeau, and Jean-Louis Deneubourg. The Origin of Nest Complexity in Social Insects. Complexity, Vol. 3, Issue 6, pp: 15-25, 1998.

[23] Werfel, J., K. Petersen, and R. Nagpal, 2014. Designing Collective Behavior in a Termite-Inspired Robot Construction Team. Science, Vol. 343 (6172), pp: 754-758.

[24] Petersen, Kirstin, Radhika Nagpal, and Justin Werfel. TERMES: An autonomous robotic system for three-dimensional collective construction. Robotics: Science and Systems Conference, Los Angeles, USA, 2011.

[25] Werfel, Justin, Kirstin Petersen, and Radhika Nagpal. Distributed Multi-Robot Algorithms for the TERMES 3D Collective Construction System. Modular Robotics Workshop, Intl. Conference on Robots and Systems, 2011.

[26] Petersen, Kirstin, Nils Napp, Jao-Ke Chin-Lee, Justin Werfel, and Radhika Nagpal. 3D Tracking of Building Processes in Macrotermes. VAIB workshop, Intl. Conference on Pattern Recognition, 2012.

[27] Petersen, Kirstin, Paul Bardunias, Nils Napp, Justin Werfel, Radhika Nagpal, and J. Scott Turner. Arrestant property of recently manipulated soil on *Macrotermes michaelseni* as determined through visual tracking and automatic labeling of individual termite behaviors. In submission with Journal of Behavioral Process, 2014.

[28] Brooks, Rodney A. and Anita M. Flynn. Fast Cheap and Out of Control: A Robot Invasion of the Solar System. Journal of the British Interplanetary Society, Vol 42, pp 478-485, 1989.

[29] Bignell, David E., Yves Roisin, and Nathan Lo (eds). Biology of Termites: a Modern Synthesis. 2nd edition, XIV, 2011.

[30] Pfeifer, Rolf, Max Lungarella, and Fumiya Iida. Self-Organization, Embodiment, and Biologically Inspired Robotics. Science, Vol. 318, pp: 1088-1093, 2007.

[31] Petersen, K. and J. Hallam. Autonomous Construction of Temporary Human Habitats. M.Sc. thesis from University of Southern Denmark, 2008.

[32] Broad Sustainable Building, Broad Group, Changsha, China.

[33] Ikeda, Yuichi and Tsunenori Harada. Application of the Automated Building Construction System using the Conventional Construction Method Together. Intl. Symposium of Automation and Robotics in Construction, 2006.

[34] Acuna, Maria Isabel. Reducing Time in the Construction of High Rise Buildings. Thesis with Civil and Environmental Engineering at Massachusetts Institute of Technology, 2000.

[35] Construction-Robotics, http://construction-robotics.com

[36] Stonespray.com

[37] Galloway, Kevin C., Rekha Jois and Mark Yim. Factory Floor: A Robotically Reconfigurable Construction Platform. In: Proceedings of 2010 IEEE Intl. Conference on Robotics and Automation, Anchorage, Alaska, USA, 2010.

[38] Napp, Nils, and Eric Klavins. Robust by Composition: Programs for Multi-Robot Systems. In: Proceedings of 2010 IEEE Intl. Conference on Robotics and Automation. Alaska, USA, 2010.

[39] Lindsey, Quentin, Daniel Melinger, and Vijay Kumar. Construction of Cubic Structures with Quadrotor Teams. In Procedings of Robotics: Science and Systems. Los Angeles, CA, USA, 2011.

[40] Vicon Motion Systems, Inc. http://www.vicon.com.

[41] Yun, Seung-kook, David Alan Hjelle, Eric Schweikardt, Hod Lipson and Daniela Rus. Planning Reconfiguration of Grounded Truss Structures with Truss Climbing Robots that Carry Truss Elements. IEEE Intl. Conference on Robotics and Automation, Kobe, Japan, 2009.

[42] Hjelle, David, and Hod Lipson. A Robotically Reconfigurable Truss. In: Proceedings of ASME/IFToMM Intl. Conference on Reconfigurable Mechanisms and Robots, 2009.

[43] Bolger, Adrienne, Matt Faulkner, David Stein, Lauren White, Seung-kook Yun and Daniela Rus. Experiments in Decentralized Robot Construction with Tool Delivery and Assembly Robots. IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, Taipei, Taiwan, 2010.

[44] Worcester, James, Joshua Rogoff and M. Ani Hsieh. Constrained Task Partitioning for Distributed Assembly. IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 2011.

[45] Bishop, J., S. Burden, E. Klavins, R. Kreisberg, W. Malone, N. Napp, and T. Nguyen. Programmable Parts: A Demonstration of the Grammatical Approach to Self-Organization. Intl. Conference on Intelligent Robots and Systems, pp: 3684-3691, IEEE/RSJ Robotics and Automation Society, 2005.

[46] Klavins, Eric, Robert Ghrist, and David Lipsky. A Grammatical Approach to Self-Organizing Robotic Systems. IEEE Transactions on Automatic Control, Vol. 51, No. 6, pp: 949-962, 2006.

[47] Matthey, Loic, Spring Berman and Vijay Kumar. Stochastic Strategies for a Swarm Robotic Assembly System. IEEE Intl. Conference on Robotics and Automation, Kobe, Japan, 2009.

[48] Wismer, Stefan, Gregory Hitz, Michael Bonani, Alexey Gribovskiy and Stephane Magnenat. Autonomous Construction of a Roofed Structure: Synthesizing Planning and Stigmergy on a Mobile Robot. IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, Algarve, Portugal, 2012.

[49] Napp, Nils and Radhika Nagpal. Distributed Amorphous Ramp Construction in Unstructured Environments. Robotica, 2014.

[50] Everist, Jacob, Kasra Mogharei, Harshit Suri, Nadeesha Ranasinghee, Berok Khoshnevis, Peter Will, and Wei-Min Shen. A System for In-Space Assembly. IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, Sendal, Japan, 2004.

[51] Werfel, Justin, Yaneer Bar-Yam, Daniela Rus, and Radhika Nagpal. Distributed Construction by Mobile Robots with Enhanced Building Blocks. IEEE Intl. Conference on Robotics and Automation, 2006.

[52] Terada, Yuzuru and Satoshi Murata. Automatic Modular Assembly System and its Distributed Control. Intl. Journal of Robotics Research, Vol. 27, No. 3-4, pp: 445-462, 2008.

[53] Terada, Yuzuru, and Satoshi Murata. Automatic Assembly System for a Large-Scale Modular Structure: Hardware design of module and assembler robot. In: Proceedings of 2004 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, pp: 2349-2355, Sendai, Japan, 2004.

[54] Groβ, Roderich and Marco Dorigo. Self-Assembly at the Macroscopic Scale. Proceedings of the IEEE, Vol. 96, No. 9, pp: 1490-1508, 2008.

[55] Yoon, Yeoreum, and Daniela Rus. Shady3D: A Robot that Climbs 3D Trusses. IEEE Intl. Conference of Robotics and Automation, Roma, Italy, 2007.

[56] Gerkey, Brian P. and Maja J. Mataric. Pusher-watcher: An approach to fault-tolerant tightly coupled robot coordination. In: Proceedings of 2002 IEEE Intl. Conference on Robotics and Automation, pp: 464-469, Washington D.C., USA, 2002.

[57] Melhuish, Chris, Jason Welsby, and Charles Edwards. Using Templates for Defensive Wall Building with Autonomous Mobile Ant-Like Robots. In Proceedings of Towards Intelligent Autonomous Mobile Robots 99, UK 1999.

[58] Sellner, Brennan, Frederik W. Heger, Laura M. Hiatt, Reid Simmons, and Sanjiv Singh. Coordinated Multi-Agent Teams and Sliding Autonomy for Large-Scale Assembly. Proceedings of the IEEE 94, pp: 1425-1444, 2006.

[59] Karsai, Istvan, and Zsolt Penzes. Comb Building in Social Wasps: Self-organization and Stigmergic Script. Journal of theoretical Biology 161, pp: 505-525, 1993.

[60] Mason, Zachary. Programming with stigmergy: using swarms for construction. In Proceedings of Artificial Life VIII, pp: 371-374, Sydney, Australia 2002.

[61] Bonabeau, Eric, Guy Theraulaz, Jean-Louis Deneubourg, Nigel R. Franks, Oliver Rafelsberger, Jean-Louis Joly, Stephane Blanco. A model for the emergence of pillars, walls and royal chambers in termite nests. Philosophical Transactions: Biological Sciences, Vol. 353, No. 1375, pp: 1561-1576, 1998.

[62] Ladley, Dan and Seth Bullock. The role of logistic constraints in termite construction of chambers and tunnels. Journal of Theoretical Biology, Vol. 234, pp: 551-564, 2005.

[63] Khuong, Anaïs, Guy Theraulaz, Christian Jost, Andrea Perna, and Jacques Gautrais. A computational model of ant nest morphogenesis. In Advances in Artificial Life, ECAL 2011: Proceedings of the Eleventh European Conference on the Synthesis and Simulation of Living Systems, 2011.

[64] von Mammen, Sebastian, Christian Jacob, and Gabriella Kokai. Evolving Swarms that Build 3D Structures. In Proceedings of IEEE Congress on Evolutionary Computation, Vol. 2, pp: 1434-1441, Edinburgh, UK, 2005.

[65] Bullock, Seth, Dan Ladley, and Michael Kerby. Wasps, Termites, and Waspmites: Distinguishing Competence from Performance in Collective Construction. Artificial Life, Vol. 18, pp: 267-290, 2012.

[66] Parker, Chris A. C., Hong Zhang, and C. Ronald Kube. Blind Bulldozing: Multiple Robot Nest Construction. Proceedings of the IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, Las Vegas, Nevada, USA, 2003.

[67] Wawerla, Jens, Guarav S. Sukhatme, and Maja J. Mataric. Collective Construction with Multiple Robots. In Proceedings of IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, Lausanne, Switzerland, 2002.

[68] Stewart, Robert L. and R. Andrew Russell. A Distributed Feedback Mechanism to Regulate Wall Construction by a Robotic Swarm. Adaptive Behavior, Vol. 14, No. 1, pp: 21-51, 2006.

[69] Beni, Gerardo. From Swarm Intelligence to Swarm Robotics. In Swarm Robotics, Lecture Notes in Computer Science, Vol. 3342, pp: 1-9, 2005.

[70] Barca, Jan Carlo, and Y. Ahmet Sekercioglu. Swarm robotics reviewed. Robotica, Vol. 31, Issue 3, pp: 345-359, 2013.

[71] Neuman, B. Clifford. Scale in Distributed Systems. Information Sciences Institute, University of Southern California, 1994.

[72] Russell, S. and P. Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall, Upper Saddle River, NJ, 2009.

[73] Dollar, Aaron M. and Robert D. Howe. Joint Coupling Design of Underactuated Grippers. Proceedings of ASME Intl. Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE), 2006.

[74] Sangbae, Kim, M. Spenko, S. Trujillo, B. Heyneman, D. Santos, and M. R. Cutkosky. Smooth Vertical Surface Climbing with Direction Adhesion. Published in Robotics, IEEE Transactions, Vol. 24, Issue 1, pp: 65-74, 2008.

[75] Sangbae, Kim, A. T. Asbeck, M.R. CutkostkySinybotII: climbing hard walls with compliant microspines. Proceedings of the 12th Intl. Conference on Advanced Robotics (ICAR), pp: 601-606, 2005.

[76] Prahlad, H., R Pelrine, S. Stanford, J. Marlow and R. Kornbluh. Electroadhesive robots – wall climbing robots enabled by a novel, robust, and electrically controllable adhesion technology. IEEE Intl. Conference on Robotics and Automation (ICRA), pp: 3028-3033, 2008.

[77] Wile, Gregory and Dean M. Aslam. Design, Fabrication and Testing of a Miniature Wall Climbing Robot Using Smart Robotic Feet. Proceedings of the Intl. Conference on Cybernetics and Information Technologies, Systems and Applications, editors M. Saavoie et al. Vol. III, pp: 87-92, 2007.

[78] Rochat, F, P. Schoeneich, O. Truong-Dat Nguyen and F. Mondada. TRIPILLAR: miniature magnetic caterpillar climbing robot with place transition ability. Intl. Conference on Climbing and Walking Robots (CLAWAR), 2009.

[79] Lam, Tin Lun, and Yangsheng Xu. A flexible tree climbing robot: Treebot – design and implementation. IEEE Intl. Conference on Robotics and Automation (ICRA), pp: 5849-5854, 2011.

[80] Brewer, T, K. N. Kaipa, and S. K. Gupta. A Quadruped Robot with On-Boarding Sensing and Parameterized Gait for Stair Climbing. 15th Intl. Conference on Climbing and Walking Robots (CLAWAR), 2012.

[81] Martin Buehler. Dynamic Locomotion with One, Four and Six-Legged Robots. McGill Research Centre for Intelligent Machines, Montreal, 2005.

[82] Estier, T., Y. Crausaz, B. Merminod, M. Lauria, R. Piguet, and R. Siegwart. An Innovative Space Rover with Extended Climbing Abilities. Robotics 2000: pp. 333-339, 2000.

[83] Eich, M, F. Grimminger, and F. Kirchner. A versatile Stair-Climbing Robot for Search and Rescue Applications. IEEE Intl. Workshop on Safety, Security and Rescue Robotics (SSRR), pp: 35-40, 2008.

[84] Moore, E. Z., D. Campbell, F. Grimminger, and M. Buehler. Reliable Stair Climbing in the Simple Hexapod 'RHex'. In Proceedings of the IEEE Intl. Conference on Robotics and Automation (ICRA), Vol. 3, pp: 2222-2227, 2002.

[85] Quinn, Roger D., Gabrial M. Nelson, Richard J. Bachmann, Daniel A. Kingsley, John Offi, and Roy E. Ritzmann. Insect Designs for Improved Robot Mobility. Proc. of Intl. Conference on Climbing and Walking Robots (*CLAWAR*), Berns and Dillmann eds., Prof. Eng. Pub., pp: 69-76, 2001.

[86] Mourikis, Anastasios I., Nikolas Trawny, Stergios I. Roumeliotis, Daniel M. Helmick, and Larry Matthies. Autonomous Stair Climbing for Tracked Vehicles. Intl. Journal of Robotics Research, Vol. 26, no. 7, pp: 737-758, 2007.

[87] Bicchi, Antonio and Vijay Kumar. Robotic Grasping and Contact: A Review. IEEE Intl. Conference on Robotics and Automation (ICRA), 2000.

[88] Gilpin, Kyle, Ara Knaian, and Daniela Rus. Robot Pebbles: One Centimeter Modules for Programmable Matter through Self-Disassembly. IEEE Intl. Conference on Robotics and Automation (ICRA), 2010.

[89] Stuart, Alastair M. Social Behavior and Communication. pp: 193-232. In Kumar Krishna and Frances M. Weesner (ed.), Biology of Termites, Vol. 1, Academic Press, N.Y. 1969.

[90] Indian Scientist Varahamihira (A.D. 505-587) in the ancient text of India Brihat Samhita (Sanskrit).

[91] Prasad, E. A. V., M Jayarama Gupta, and Colin E. Dunn. Significance of Termite Mounds In Gold Exploration. Current Science, Vol. 56, No. 23, pp: 1219-1222, 1987.

[92] Smeathman, Henry. Account of the Termites, which are found in Africa and other hot climates. Read at the Royal Society, London, February 15, 1781.

[93] Hagen, H. A. Monographie der Termiten. Linn. Entomology, Stettin., Vol. X, 1855.

[94] Bruinsma, O.H. 1977. An analysis of building behavior of the termite *Macrotermes subhyalinus*. Proc. VIII Congr. IUSSI, Wageningen.

[95] Darlington, J. P. E. C. (1990). Populations in nests of the termite *Macrotermes subhyalinus* in Kenya. *Insectes Sociaux*, *37*(2), 158–168.

[96] Hesse, P. R. A Chemical and Physical Study of the Soils of Termite Mounds in East Africa. Journal of Ecology, Vol. 43, No. 2, pp: 449-461, 1955.

[97] Wood, T. G. Termites and the soil environment. Biology and Fertility of Soils, Vol. 6, pp: 228-236, 1988.

[98] Dangerfield, J. M., T. S. McCarthy and W. N. Ellery. The mound-building termite *Macrotermes michaelseni* as an ecosystem engineer. Journal of Tropical Ecology, Vol. 14. pp: 507-520, 1998.

[99] Abe, Susumu S., Takashi Kotegawa, Taisuke Onishi, Yoshinori Watanabe, and Toshiyuki Wakatsuki. Soil particle accumulation in termite (Macrotermes bellicosus) mounds and the implications for soil particle dynamics in a tropical savanna Ultisol. Ecological Research, Vol. 27, no. 1, 1997.

[100] Wheeler, W. M. The ant-colony as an organism. Journal of Morphology, 22(2), pp: 307-325, 1911.

[101] Emerson, A. E. Social coordination and the superorganism. Am. Midland Naturalist 21, pp: 182-209, 1939.

[102] Lüscher, M. The termite and the cell. Scientific American 188, Vol. 5, pp:74-78, 1953.

[103] Turner, J. Scott. Dirt Lungs. Natural History, 2010.

[104] Lee, S.-H., P. Bardunias, N.-Y. Su, and R.-L. Yang. Behavioral response of termites to tunnel surface irregularity. Behavioral Processes 78: pp 397-400, 2008.

[105] Turner, J. Scott. Termites as models of swarm cognition. Swarm Intelligence, Vol. 5, pp: 19-43, 2011.

[106] Maeterlinck, Maurice. The Life of the White Ant. Translated by Alfred Sutro, printed by Unwin Brothers, Ltd, London 1927.

[107] Dornhaus, Anna, Jo-Anne Holley, Victoria G. Pook, Gemma Worswik, and Nigel R. Franks. Why do not all workers work? Colony size and workload during emigrations in the ant *Temnothorax albipennis*. Behavioral Ecology and Sociobiology, Vol. 63, pp: 43-51, 2008.

[108] Affolter, J. and R. H. Leuthold. Quantitative and qualitative aspects of trail pheromones in *Macrotermes subhyalinus* (Isoptera, Termitidae). Insectes Sociaux, Vol. 47, pp: 256-262, 2000.

[109] Bardunias, Paul, and Nan-Yao Su. Tunnel orientation by workers of *Coptotermes formosanus* (Isoptera: Rhinotermitidae) subjected to unilateral antennal ablation. Florida Entomology 93, pp: 310-312, 2010.

[110] Katz, Yael, Kolbjørn Tunstrøm, Christos C. Ioannou, Cristián Huepe, and Iain D. Couzin. Inferring the structure and dynamics of interactions in schooling fish. PNAS, Vol. 108, no. 46, 2011.

[111] Ohayon S, O. Avni, Taylor AL, O. Perona, Egnor SER. Automated multi-day tracking of marked mice for the analysis of social behaviour, Journal of Neuroscience Methods, Vol. 219, no. 1 , pp: 10–19, 2013.

[112] Mersch, Danielle P., Alessandro Crepsi, and Laurent Keller. Tracking Individuals Shows Spatial Fidelity Is a Key Regulator of Ant Social Organization. Science 31, Vol. 340, no. 6136, pp: 1090-1093, 2013.

[113] Kimura, T., M. Ohashi, K. Crailsheim, T. Schmickl, R. Okada, G. Radspieler, and H. Ikeno. Development of a New Method to Track Multiple Honey Bees with Complex Behaviors on a Flat Laboratory Arena. PLoS ONE 9 (1), 2014.

[114] Branson, K., A. A. Robie, J. Bender, P. Perona and M. H. Dickinson. High-throughput ethomics in large groups of *Drosophila*. Nature Methods 6, pp: 451-457, 2009.

[115] Correll, Nikolaus, Gregory Sempo, Yuri Lopez de Meneses, José Halloy, Jean-Louis Deneubourg, and Alcherio Martinoli. SwisTrack: A Tracking Tool for Multi-Unit Robotic and Biological Systems. Proceedings of the 2006 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, Beijing, China, 2006.

[116] Fletcher, Mary, Anna Dornhaus and Min C. Shin. Multiple Ant Tracking with Foreground Maximization and Variable Target Proposal Distribution. Workshop on Applications of Computer Vision (WACV), 2010.

[117] Fasciano, Thomas, Hoan Nguyen, Anna Dornhaus, and Min C. Shen. Tracking Multiple Ants in a Colony. WACV '13 Proceedings of the 2013 IEEE Workshop on Applications of Computer Vision (WACV), pp: 534-540, 2013.

[118] Mersch, Danielle P., Alessandro Crespi, and Laurent Keller. Tracking Individuals Show Spatial Fidelity is a Key Regulator of Ant Social Organization. Science Magazine, Vol. 340, pp: 1090-1093, May 2013.

[119] Khan, Z., T. Balch, and F. Dellaert. MCMC-based particle filtering for tracking a variable number of interacting targets. IEEE Transactions on Pattern Analysis and Machine Intelligence, pp: 1805–1918, 2005.

[120] Khan, Z., T. Balch, and F. Dellaert. Mcmc data association and sparse factorization updating for real time multitarget tracking with merged and multiple measurements. IEEE Transactions on Pattern Analysis and Machine Intelligence 28, pp: 1960–1972, 2006.

[121] Harris, W. V. and W. A. Sands. The social organization of termite colonies. Symposium of the Zoological Society of London 14, pp: 113-131, 1965.

[122] Stuart, A. M. Alarm, defense and construction behavior relationships in termites (Isoptera). Science 156, PP: 1123-1125, 1967.

[123] Robinson, Gene E. Regulation of Division of Labor in Insect Societies. Annual Review of Entomology, Vol. 32, pp: 635-65, 1992.

[124] Crosland, M. W. J. , C. M. Lok, T. C. Wong, M. Shakarad and J. F. A. Traniello. Division of labour in a lower termite: the majority of tasks are performed by older workers. Animal Behavior, Vol. 54, pp: 999-1012, 1997.

[125] Hinze, B. and R. H. Leuthold. Age related polyethism and activity rhythms in the nests of the termite *Macrotermes bellicosus* (Isoptera, Termitidae). Insectes Sociaux, Vol. 46. pp: 392-397, 1999.

[126] Leonard, Naomi, Tian Shen, Benjamin Nabet, Luca Scardovi, Iain Couzin, and Simon A. Levin. Decision versus compromise for animal groups in motion. PNAS, Vol. 109, No. 1, pp: 227-232, 2012.

[127] Andrew Ng. Stanford Machine Learning Class, week 2. Coursara.org, 2014.

[128] Li, H.-F. and N.-Y. Su. Buccal Manipulation of Sand Particles During Tunnel Excavation of the Formosan Subterranean Termite (Isoptera: Rhinotermitidae). Annals of the Entomological Society of America, Vol. 102, Issue 2, pp 333-338, 2009.

[129] Thorne, B. L. and M.I. Haverty. A review of intercolony, intraspecific and interspecific agonism in termites. Sociobiology 19: pp 115-145, 1991.

[130] Bell, W. J., L. M. Roth, and C. A. Nalepa. Cockroaches Ecology, Behavior, and Natural History. John Hopkins University Press, 2007.

[131] Brossut, R., P. Dubois, and J. Rigaud. Le grégarisme chez *Blaberus craniifer*: Isolement et identification de la pheromone. Journal of Insect Physiology 20: pp529-543, 1974.

[132] Lanman, D. and G. Taubin. Build your own 3D scanner: 3D photography for beginners. SIGGRAPH '09: ACM SIGGRAPH 2009 courses.

[133] DAVID® Laserscanner software: http://www.david-laserscanner.com/

[134] Matlab, 2012. MathWorks, version R2012a.

[135] MeshLab Visual Computing Lab, ISTI – CNR: http://meshlab.sourceforge.net/

[136] Rasband, W.S., ImageJ, U. S. National Institutes of Health, Bethesda, Maryland, USA, http://imagej.nih.gov/ij/, 1997-2014.

[137] Page, L., S. Brin, R. Motwani, T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[138] Kleinberg, Jon M. Authoritative Sources in a Hyperlinked Environment. Journal of the ACM, Vol. 46, No. 5, pp: 604-632, 1999.